Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

Ricardo Daniel Ramos Mendes

Smart* - Arquitetura para aquisição e tratamento de dados

Smart* - Architecture for the acquisition and processing of data

Universidade de Aveiro
2019

Departamento de Eletrónica,
Telecomunicações e Informática

Ricardo Daniel Ramos Mendes

Smart* - Arquitetura para aquisição e tratamento de dados

Smart* - Architecture for the acquisition and processing of data

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Cláudio Teixeira, Professor Equiparado a Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Joaquim Sousa Pinto, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus amigos pelo apoio incansável e à minha família por todo o esforço despendido que possibilitou a minha formação académica.

o júri / the jury

presidente / president

Prof. Doutor Ilídio Fernando de Castro Oliveira
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Prof. Doutor André Frederico Guilhoto Monteiro
Professor Auxiliar Convidado, Instituto Superior Miguel Torga

Doutor Cláudio Jorge Vieira Teixeira
Equiparado a Investigador Auxiliar, Universidade de Aveiro

agradecimentos /
acknowledgements

Ao professores Joaquim Sousa Pinto e Cláudio Teixeira por me terem aberto as portas e pela paciência! Aos meus colegas de trabalho e amigos pelo incentivo constante na conquista de novas fronteiras. A todos o meu obrigado!

Palavras Chave Agricultura, Big Data, IoT, SmartCities, WSN.

Resumo

Esta dissertação tem como principal objetivo a conceção de uma arquitetura que permita a aquisição, tratamento e visualização de dados. A aquisição poderá ter origem em cenários como monitorização de poluição, saúde, *fitness*, ou agricultura onde através de uma WSN (*Wireless Sensors Network*) são recolhidos dados e características para posterior análise. Afim de testar a arquitetura desenvolvida foi criado um cenário de agricultura de precisão onde é possível, em tempo real, a monitorização, suporte e divulgação dos dados recolhidos usando uma abordagem escalável.

Keywords Agriculture, Big Data, IoT, SmartCities, WSN

Abstract The main goal of this dissertation is to design an architecture that allows the acquisition, treatment and visualization of data. The acquisition may originate in scenarios such as pollution monitoring, health, fitness, agriculture where, through a WSN (Wireless Sensors Network) data and characteristics are collected for further analysis. In order to test the develop architecture, a precision agriculture scenario was created where it is possible to monitor, support and disseminate the data collected using a scalable approach in real time.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

M2M	Machine to Machine
TIC	Tecnologias da Informação e Comunicação
WSN	Wireless Sensor Networks
BSN	Body Sensor Network
GPS	Global Positioning System
IoT	Internet of Things
DSMS	Data Stream Management System
SDK	Software development kit
GSM	Groupe Special Mobile
UDP	User Datagram Protocol
LDR	Light Dependent Resistor
MQTT	Message Queuing Telemetry Transport
IP	Internet Protocol
CoAP	Constrained Application Protocol
DSN	Distributed Sensor Networks
DARPA	Defense Advanced Research Projects Agency
IDC	International Data Corporation
OMA	Open Mobile Alliance
NFC	Near Field Communication
RFID	Radio-Frequency IDentification
ISM	Industrial, Scientific and Medical
ECG	Eletrocardiograma
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
LwM2M	Lightweight Machine to Machine
API	Application Programming Interface
3GPP	3rd Generation Partnership Project
H2M	Human-to-Machine
XML	eXtensible Markup Language
OpenXPS	Open XML Paper Specification
OpenMTC	Open Machine Type Communications
REST	Representational State Transfer
XMPP	eXtensible Messaging and Presence Protocol
DHCP	Dynamic Host Configuration Protocol
TCP	Transmission Control Protocol
JSON	JavaScript Object Notation
UUID	Universal Unique Identifier
WWW	World Wide Web
W3C	World Wide Web Consortium
IETF	Internet Engineering Task Force
QoS	Quality of Service
GSMA	Global System for Mobile Communications Association
P2P	Peer-to-peer
IEEE	Institute of Electrical and Electronics Engineers
BLE	Bluetooth Low-Energy
WAN	Wide Area Network
PaaS	Platform as a Service
6LowPAN	IPv6 over Low power Wireless Personal Area Networks
LowPAN	Low power Wireless Personal Area Networks
IPv6	Internet Protocol version 6
RFC	Request for Comments
LAN	Local Area Network
MAC	Mandatory Access Control
OSI	Open System Interconnection
PnP	Plug and Play
IBM	International Business Machines
MsgPack	Message PackData Stream Management Systems
UNB	Ultra Narrow Band
LPWAN	Low-Power Wide-Area Network
CRUD	Create, Read, Update e Delete
ETSI	European Telecommunications Standards Institute
SCL	Service Capability Layers
GSCL	Gateway Service Capability Layers
NSCL	Network Service Capability Layers
IMS	IP Multimedia Subsystem

EPC	Evolved Packet Core
SIP	Session Initiation Protocol
RF	Radiofrequência
OTA	Over-the-Air
EULA	End-User License Agreement
USB	Universal Serial Bus
SQL	Structured Query Language
NoSQL	Not only SQL
RDBMS	Relational Database Management System
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

CAPÍTULO 1

Introdução

A IoT continua a ser uma área de enorme interesse e em constante crescimento. Constantemente surgem novas soluções, arquiteturas e implementações tanto ao nível dos serviços como ao nível das comunicações Machine to Machine (M2M), permitindo assim o aparecimento de novos modelos de negócio. Desta forma, surgiu a possibilidade de abstrair a gestão do hardware e software, ou seja, separar a rede de sensores da criação de sistemas de administração e visualização dos dados, permitindo interoperabilidade entre sensores e elasticidade na criação de serviços.

A evolução e o crescimento das comunicações impulsionaram que sejam ligados à Internet cada vez mais dispositivos. Dispositivos como smartphones, tablets, televisores, frigoríficos e outros dispositivos domésticos podem ser ligados à Internet e, portanto, trocar informações com o resto do mundo. Esta evolução tem chegado a dispositivos de pequenas dimensões como sensores e atuadores, transformando-os em dispositivos ligados e, potencialmente, em coisas inteligentes.

O surgimento destas coisas inteligentes ou objetos inteligentes potenciou o crescimento do conceito IoT, proporcionando uma aproximação do indivíduo ao mundo digital através da recolha de dados no mundo físico. Assim sendo estas coisas permitem a criação de "serviços" do mundo real através da sua capacidade de fornecer dados em tempo quase real. No entanto, esta capacidade para a geração e recolha de dados muitas vezes, sem o contexto apropriado, não tem qualquer significado.

Para abordar o problema da interligação e interpretação dos dados provenientes destes sensores, começaram a surgir plataformas que permitem a interoperabilidade e a catalogação dos dados. Além disso, permitem que outras aplicações e serviços consigam operar sobre os dados recolhidos e assim potenciar a criação de informação.

Com o crescimento do número de dispositivos ligados será possível otimizar a indústria, saúde, ambiente e até melhorar a qualidade de vida. Para que tal aconteça, é fundamental uma aquisição e interpretação correta desses dados, bem como a compilação de informações de múltiplas origens para a obtenção de mais resultados e assim potenciar ainda mais a IoT.

1.1 Enquadramento

Graças à evolução movida pelo avanço nas Tecnologias da Informação e Comunicação (TIC), as pessoas e coisas que as rodeiam estão cada vez mais interconectadas [1]. Novos dispositivos com extensas capacidades produzem cada vez mais dados verificando-se uma crescente necessidade de armazenar um maior e mais variado volume de dados.

As Wireless Sensor Networks (WSN) [2] são redes compostas por vários nós equipados com diversos sensores que comunicam entre eles. Estas redes pretendem, de uma forma distribuída, monitorizar características através dos seus sensores e podem ser utilizadas nos mais diversos domínios como monitorização de poluição, terramotos, agricultura e até na recolha de dados no âmbito da saúde.

Um caso particular de WSN são as Body Sensor Network (BSN) [3]. Estas têm vindo a ser utilizadas para monitorização de atividades humanas como desempenho desportivo, recuperação de lesões ou simplesmente monitorização do sono de um indivíduo. Estas redes, através dos seus *Sensor Nodes* pretendem obter várias características como temperatura, humidade, movimento, frequência cardíaca e localização de um indivíduo. Para tal, os nós da rede estão, normalmente, munidos de acelerómetros, giroscópios, bússolas, Global Positioning System (GPS) e sensores de temperatura. Dependendo das taxas de amostragem os sensores geram uma maior quantidade de dados e informação que terão de ser monitorizados e analisados *à posteriori*. Porém, muitas dessas características podem ser retiradas através de uma observação em tempo real das leituras dos diversos sensores [4]. Na agricultura, estas redes de sensores são utilizadas para a obtenção de parâmetros que permitam verificar a saúde das culturas e assim prever possíveis epidemias e ataques que prejudiquem o seu normal crescimento. Nas *Smart Cities* vários parâmetros são controlados, gerando informações essenciais para os decisores de uma cidade [5].

Tem havido um interesse crescente por partes de diversas entidades para que crianças, adultos e até idosos acompanhem os seus sinais vitais permitindo melhorar a sua saúde e consequentemente melhorar o estilo de vida. Em cenários de cuidados continuados de saúde através de uma BSN, o profissional de saúde acompanha e pode atuar sobre alguns alertas despoletados pelo sistema. Desta forma, é possível através de uma BSN recolher uma quantidade de dados significativa, processá-los em tempo real [6] e armazená-los em repositórios de dados médicos remotos para análises *offline*.

No ramo da agricultura, a introdução de sensores numa colheita ajudará o responsável na tomada de decisões e, através de alarmística, poderá ser notificado na ocorrência de alguma situação indevida. Estes cenários implicam a transmissão, armazenamento e análise de uma enorme quantidade de dados.

1.2 Motivação

A IoT tem crescido nos últimos anos e movimentará valores avultados de dinheiro daqui para a frente [7], [8]. A Cisco estima que em 2020 existirão 50 mil milhões de dispositivos ativos e estarão presentes nas mais variadas indústrias, desde a indústria da manufatura até à área

do retalho e dos serviços financeiros [9] [10]. Por outro lado, a Gartner, líder mundial em pesquisa e consultoria em tecnologias da informação, afirmou em dezembro de 2013 que a IoT crescerá para 26 mil milhões de unidades ativas em 2020 [8]. Estimando-se que a própria população crescerá até 7,6 mil milhões de pessoas, no que resultará em 6,58 dispositivos por pessoa [9].

Ainda, a Boston Consulting Group, estima que se gastarão cerca de 250 mil milhões de euros em 2020 em IoT e, todas as indústrias deverão aumentar a sua aposta nesta área [11]. Por fim, a Gartner estima que em 2017, de todas as unidades IoT instaladas (cerca de 8380.6 milhões), aproximadamente 63% (5244.3 milhões) são associadas ao consumidor e, assim continuará até 2020 [12].

Na Figura 1.1 e na Tabela 1.1 encontram-se a estimativa de gastos e a estimativa de unidades de IoT, respetivamente.

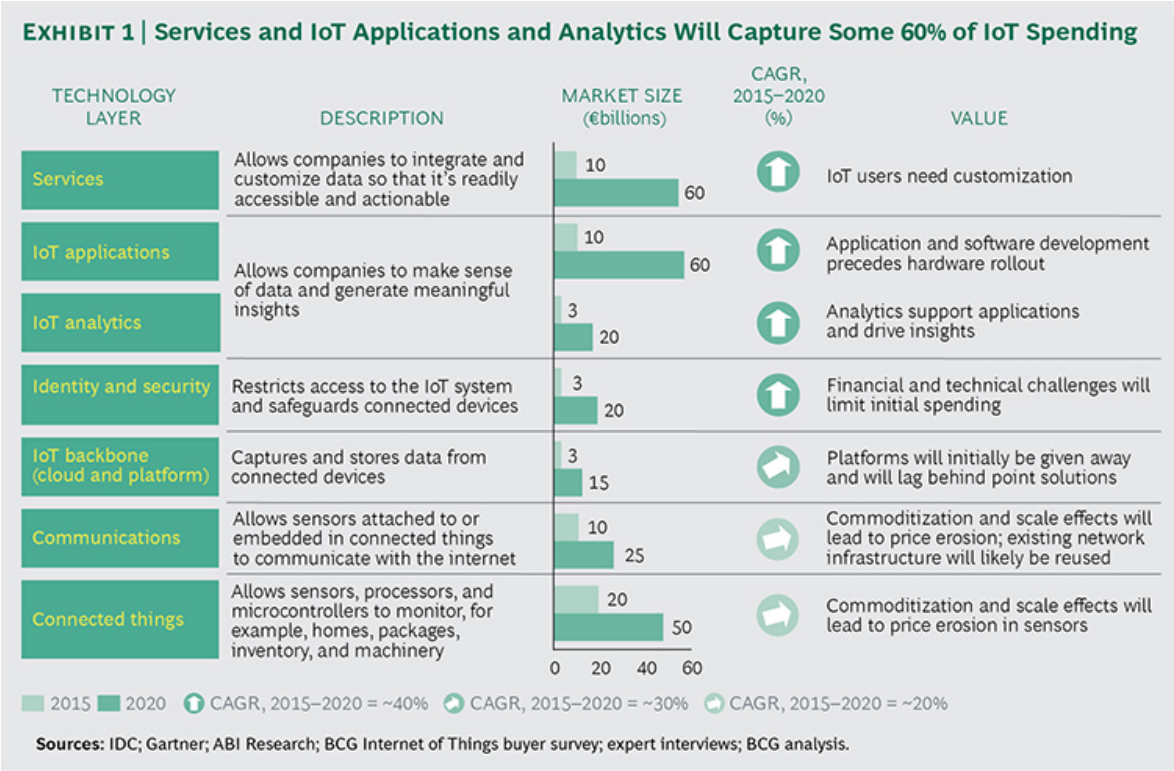

Figura 1.1: Estimativa de gastos em IoT nas diferentes áreas de negócio[11]

Ao conectar tudo à *Internet* passará a existir uma quantidade inimaginável de oportunidades de negócios onde setores como a indústria, logística e saúde são algumas das principais áreas de interesse na IoT [13].

As BSN são redes criadas a partir da interligação de pequenos sensores que são utilizados juntos do corpo humano e permitem monitorizar diversas atividades. Os diversos sensores recolhem dados fisiológicos e enviam para uma unidade central denominada de *Coordinator* onde por vezes existe processamento para extração de informações e características ou simplesmente reencaminha para uma unidade de processamento externa. Neste último caso, os dados são transferidos instantaneamente por exemplo para um médico que conseguirá

acompanhar um paciente em tempo real e tomar decisões tendo por base esses dados [14].

Categoria	**2016**	**2017**	**2018**	**2020**
Consumidor	3963.0	5244.3	7036.3	12863.0
Negócio: Industria transversal	1102.1	1501.0	2132.6	4381.4
Negócio: Industria específica	1316.6	1635.4	2027.7	3171.0
Total	**6381.8**	**8380.6**	**11196.6**	**20415.4**

Tabela 1.1: Estimativa do número de dispositivos IoT ao longo dos próximos anos[12]

Atualmente estas redes estão a ser utilizadas para reconhecimento de atividades através de acelerómetros e giroscópios, deteção precoce de ataques cardíacos verificando alterações nas leituras do Eletrocardiograma (ECG) e aplicação de autoinjeções de insulina num paciente. Áreas como consolas de vídeo jogos têm apostado em aplicações de fitness onde são extraídos gastos calóricos, postura e batimentos cardíacos, o que tornou possível o surgimento de aplicações de reabilitação motora. Através de BSN são detetadas quedas, é possível a mitigação de doenças como Parkinson, diabetes e asma o que tem sido uma mais valia para a gerontologia. Noutras áreas que não a saúde, estas redes de sensores são capazes de realizar leituras de vários parâmetros tendo como origem um dispositivo e controlar atuadores para a aplicação de fitofármacos como no caso de produções hidropónicas.

Com a crescente utilização destas redes de sensores, potencialmente serão criadas novas áreas de negócio ou simplesmente serão aperfeiçoadas as já existentes. Também os modelos de negócio associados irão sofrer alterações adaptando-se aos novos desafios que vão sendo criados.

1.3 Problema

Atualmente, existem diversas alternativas no mercado para a interligação de sensores seja em WSN ou noutras redes de sensores. Atualmente estas plataformas surgem associadas à venda desses sensores, sendo que maioritariamente estas plataformas são proprietárias, com poucas capacidades de personalização e integração de outros sistemas por parte dos utilizadores. Desta forma é de extrema importância dar liberdade ao utilizador para escolher que sensores pretende utilizar e assim sendo, é necessário garantir uma total integração dos vários sensores no mercado.

Por outro lado, dependendo da utilização a que estas redes de sensores se destinem poderá ser necessário lidar com enormes quantidades de dados e frequências de amostragem extremamente elevadas, obrigando a uma procura de alternativas que permitam lidar com essas necessidades e que não impliquem um investimento elevado.

Por fim, a IoT sofre de problemas de fragmentação, interoperabilidade e escalabilidade. Existem protocolos e padrões definidos especificamente para o IoT, mas não estão a ser adotados por todos os arquitetos de sistemas IoT, o que leva a problemas de interoperabilidade. Para

que a IoT cresça até ao seu pleno potencial, estas e outras preocupações terão de ser resolvidas o quanto antes para que seja possível a adoção destas redes de sensores.

1.4 Contribuição

Através desta dissertação pretende-se estudar o desafio de *Big Data* associado a uma WSN e outras redes de sensores, verificando arquiteturas e plataformas existentes, analisando o potencial e as limitações de cada uma. A análise das arquiteturas já existentes tem como principal objetivo identificar as suas principais limitações de forma a ser possível propor uma nova arquitetura. A nova arquitetura enquadra-se na utilização de redes de sensores e terá como principal objetivo a distribuição e tratamento de dados provenientes de vários dispositivos para visualização em tempo real ou futura análise dos mesmos. Nesta arquitetura também está contemplada a distribuição dos dados entre várias visualizações e permite ao utilizador definir como pretende receber, manipular e visualizar os dados.

Para além do foco na arquitetura, espera-se ainda implementar um sistema no âmbito da agricultura de precisão, tirando partido da arquitetura proposta.

1.5 Metodologia de investigação e desenvolvimento

Ao longo do desenvolvimento desta dissertação foi utilizada a metodologia *The Engineering Design Process*[15]. Esta assenta numa série de passos seguidos iterativamente até à resolução do problema. Inicialmente é definido o problema, investigado o tema e determinado os requisitos. De seguida através de *brainstorming*, é escolhida a melhor solução. Esta segue para desenvolvimento, criando um protótipo, e de seguida são realizados os testes à solução encontrada. Por fim, se a solução não for de encontro com os requisitos estabelecidos, o processo é realizado novamente[15].

Seguindo os passos da metodologia adotada, o problema em questão foi definido pela equipa. Inicialmente foi definido o problema, de seguida foram recolhidos e debatidos os dados sobre a temática. Ao longo do desenvolvimento da solução e protótipo, tiveram-se em consideração a documentação, código, bem como a sua integração num repositório de dados (*GitHub*). Desta forma, o desenvolvimento foi documentado a cada passo, prevendo que a pesquisa de detalhes de implementação possa ser realizada posteriormente com maior facilidade.

Após os testes e avaliação da solução criada, foram analisados os resultados para verificar se estavam de acordo com o pretendido. Ao longo desta dissertação foram realizadas várias iterações afim de melhorar a solução proposta.

1.6 Estrutura da dissertação

Esta dissertação está dividida em cinco capítulos que serão caracterizados de seguida. O **Capítulo 1** apresenta a introdução ao trabalho, mostrando o seu enquadramento, motivação e identificando o problema de investigação. No **Capítulo 2** serão descritos alguns casos de

estudo de plataformas de distribuição de dados provenientes de redes de sensores como é o caso de WSN, apresentando assim o estado de arte.

O **Capítulo 3** fornece uma visão completa sobre a arquitetura proposta, descrevendo detalhadamente cada componente e como tudo se encaixa.

Durante o **Capítulo 4** será validada a arquitetura proposta, fornecendo uma explicação detalhada sobre a implementação, as tecnologias usadas e os fluxos que ela suporta.

Por fim, no **Capítulo 5** serão apresentadas as conclusões do trabalho realizado, sistematizando o trabalho e apresentando propostas de trabalho futuro.

CAPÍTULO 2

Estado de Arte

À medida que o conceito da IoT foi surgindo, as aplicações comerciais e industriais começaram a adotar diferentes tecnologias e protocolos nas suas soluções. A maioria dessas soluções IoT, tipicamente proprietárias e bastante específicas, transmitem os dados para um *gateway* que após agregar os vários dados recolhidos, os envia para a *cloud*. Desta forma não existe uma ligação Peer-to-peer (P2P) entre os diferentes dispositivos e a *internet*. De uma maneira geral, esses sistemas possuem limitações em termos de interoperabilidade, acessibilidade e escalabilidade.

No entanto, grandes entidades, como Internet Engineering Task Force (IETF) e Institute of Electrical and Electronics Engineers (IEEE), reconheceram esse problema e começaram a desenvolver padrões e protocolos projetados especificamente para soluções IoT. Através dos seus esforços, atualmente, é possível conectar os diferentes nós diretamente à *internet*. Assim fornecem recursos para gestão dos dispositivos, interfaces padrão de M2M e comunicação direta com serviços e aplicações na *internet*. Tudo isso resolve a maioria dos problemas das soluções atuais: acessibilidade, interoperabilidade, escalabilidade e segurança.

Este capítulo apresenta os esforços que têm sido realizados pelas principais entidades para desenvolver uma arquitetura padrão para sistemas M2M e IoT. Em poucas palavras, todos esses protocolos e padrões funcionam para o mesmo fim: implementar uma arquitetura de P2P interoperável, independentemente da camada de transporte e das tecnologias subjacentes. Além desses protocolos e padrões, também existem arquiteturas relevantes e necessárias que combinam todos os serviços presentes num sistema IoT e M2M permitindo a interoperabilidade. Capacidades como gestão de dispositivos, *bootstrapping*, interfaces padrão são requisitos nos sistemas atuais de IoT e M2M. Do ponto de vista da indústria, isso significa simplesmente que permitirá uma implantação mais rápida e confiável de uma solução M2M.

As próximas secções detalharão os padrões e propostas mais recentes para uma arquitetura no âmbito da IoT.

2.1 Contextualização

Nos últimos anos, tem havido um número crescente de plataformas distribuídas baseadas em BSN para o desenvolvimento de aplicações no domínio da saúde. Muitas destas plataformas utilizam os diversos sensores para acompanhar parâmetros vitais de um doente. No âmbito da agricultura de precisão começam a surgir plataformas de suporte à decisão para colheitas onde o acompanhamento constante é vital para obtenção dos melhores resultados. Nas *Smart Cities*, câmaras municipais e outras entidades oficiais apoiam as suas decisões tendo por base dados resultantes de sensores espalhados nas cidades.

Vários têm sido os esforços para criar sistemas de gestão de fluxo de dados denominados de Data Stream Management System (DSMS) [16]. Estes são sistemas projetados para lidar com grandes quantidades de dados e informação como é o caso de observações de sensores, onde num curto espaço de tempo são gerados grandes volumes de dados dependendo da precisão pretendida. Alguns dos sistemas atuais são já capazes de processar todos esses dados em tempo quase real a fim de retirar algumas características a partir da leitura dos diversos sensores.

Todos estes sistemas enfrentam enormes quantidades de dados provenientes de vários sensores e a fragmentação, interoperabilidade e escalabilidade desses mesmos sistemas são sem dúvida um enorme desafio.

2.2 Wireless Sensor Networks

Uma WSN é uma rede de inúmeros sensores, espalhados num ambiente autónomo e capazes de recolher e transmitir dados do mundo físico utilizando comunicações sem fio [17]. Um dispositivo pode agregar vários sensores e atuadores para poder interagir com o mundo físico.

O termo WSN remonta ao ano de 1980 através do programa Distributed Sensor Networks (DSN) da Defense Advanced Research Projects Agency (DARPA). As DSN são compostas por vários nós distribuídos espacialmente, sendo que cada nó tem um baixo custo e funciona autonomamente, mas em colaboração com os restantes nós [18]. Nestes ambientes o processamento era realizado em minicomputadores pois a *Ethernet* começava a dar os primeiros passos na sua disseminação [18].

Através de uma WSN é possível monitorizar variáveis de ambiente como temperatura, humidade, pressão, entre outras, mas também seguir animais, vigiar florestas, detetar inundações e realizar previsões meteorológicas com maior precisão [19]. Afim de entender melhor as mudanças climáticas num ambiente glaciar, investigadores da Universidade de Southampton construíram um sistema de monitorização para averiguar a influência das alterações do nível do mar no aquecimento global. [20]. Foram colocados sensores no interior do gelo e através de uma WSN foi possível reduzir o impacto que a colocação de cabos provocaria no meio ambiente.

2.3 Machine to Machine

M2M refere-se a tecnologias que permitem a comunicação entre dispositivos através de redes de comunicação com ou sem fios, tudo sem intervenção humana.

Hoje, o conceito de M2M vai um pouco mais longe conseguindo que máquinas comuniquem entre si, mas que também sejam capazes de dar sentido aos dados que recebem. O conceito de comunicações entre máquinas, como *routers* e servidores, ou sistemas de telemetria não é novo. Isso abriu um mundo novo de oportunidades, no qual as máquinas nos ajudarão a entender melhor o mundo em que vivemos.

O progresso tecnológico nas últimas décadas, a redução dos custos dos componentes eletrónicos, a adoção do Internet Protocol (IP) e a disseminação da *internet* ligando todo o tipo de dispositivos fez com que houvesse uma evolução no termo M2M [21].

Os custos decrescentes dos componentes eletrónicos, bem como a redução das suas dimensões, levaram a uma proliferação destes pequenos dispositivos. Sensores e atuadores outrora de grandes dimensões e caros podem ser encontrados com dimensões de milímetros ou até mesmo de nanómetros e a preços reduzidos[22]. Isso permitiu a implantação de milhares de sensores e atuadores em vários cenários, criando assim uma oportunidade sem precedentes para uma melhor perceção do mundo físico.

Além disso, a crescente adoção do IP, bem como novos protocolos de transporte, como o Constrained Application Protocol (CoAP), orientado para dispositivos menos capazes, permitiu a criação de maiores redes de sensores em ambientes com e sem fio.

Estas redes permitem uma melhor compreensão do que nos rodeia e permitem a integração desses dispositivos com eletrodomésticos, como frigoríficos e máquinas de café, tornando-os mais inteligentes e capazes de tomar decisões. O meio ambiente, a saúde e a indústria são ambientes propícios para a utilização deste tipo de redes.

Com o avanço das redes de telecomunicações sem fios criou-se um novo paradigma de dispositivo sempre ligado, sendo esses dispositivos smartphones, tablets e smartwatches, apelidados de *smart devices*. Esta evolução dos dispositivos permitiu que estes se ligassem à *internet* e a M2M, prosperassem e inevitavelmente a IoT crescesse.

2.4 M2M até IoT

Neste momento o panorama da IoT ainda é bastante indefinido, estando universidades e empresas a implementar soluções da maneira que eles consideram corretas, mas por vezes a interoperabilidade é esquecida[21].

Num cenário onde todas as casas estariam munidas de sensores que permitissem verificar o nível dos depósitos de lixo seria possível à empresa de recolha de lixo planear a recolha de uma forma mais eficiente, mas para que tal seja possível seria necessária uma mudança de paradigma e aí a interoperabilidade na IoT é fundamental.

Para que as "coisas" comuniquem entre si, antes de mais, é necessário que elas saibam como o fazer. Isso significa que têm de existir interfaces comuns a que os dispositivos devam obedecer para que a interoperabilidade entre sistemas seja possível. Porém, o que acontece

hoje em dia é que grupos de empresas têm visões diferentes sobre o que cada interface deve ter. Diferentes consórcios estão a desenhar padrões diferentes que, muita vezes, não são compatíveis.

Existem esforços por algumas empresas para que a IoT siga para um próximo nível. Empresas como Amazon, Apple, AT&T, Cisco, General Electric, Google, International Business Machines (IBM), entre outras, são importantes para essa mudança, graças à sua influência sobre o mercado.

A IBM através da plataforma Watson, onde através da recolha e tratamento de enormes quantidades de dados é capaz de superar a cognição humana [23]. A AT&T prevê que a IoT permitirá naturalmente evoluir a empresa pois muitas destas redes de sensores dependem de ligações de banda larga sem fios para que dispositivos e plataformas comuniquem entre si [24]. A Cisco, que tem uma forte presença no mercado em soluções de *software* e *hardware* começou a concentrar os seus esforços em parceria com a Ericson para a criação de soluções IoT [25]. A Amazon através do seu assistente de casa *Echo* pode controlar vários dispositivos inteligentes presentes numa casa, tornando-se assim o centro de uma casa inteligente.

2.5 Internet of Things

O conceito de IoT, apesar de não ter uma definição universal, representa a interligação de objetos quotidianos com capacidade para detetar, processar e transmitir informações entre dispositivos, através da *internet*, sem existir obrigatoriamente interação humana[26]. A proximidade desta definição com a definição de M2M por vezes gera confusão. Porém, é a tecnologia M2M que torna possível a existência da IoT, mas tem sido a crescente adoção da IoT que tem permitido a evolução da M2M.

Um estudo da Machina Research estima que a receita associada a M2M em 2022 atingirá os $1.2 biliões face aos $200 mil milhões em 2013. Além disso, estudos da Global System for Mobile Communications Association (GSMA) indicam que a M2M é responsável por $\frac{1}{10}$ das ligações móveis nos EUA e $\frac{1}{20}$ na Oceânia e Europa [27]. Sem dúvida que a M2M é uma área com enormes oportunidades de negócio, investimento e em clara expansão. Alguns países têm vindo a alterar as suas legislações, incentivando a adoção de sistemas M2M, como por exemplo na Suécia, onde $\frac{1}{4}$ das ligações móveis são provenientes de dispositivos M2M [28]. Tal rácio deve-se a uma alteração legislativa onde foi estabelecido que cada família deveria monitorizar mensalmente os gastos elétricos através de contadores inteligentes.

Com o crescente interesse da IoT(Figura 2.1) e M2M existem mais e maiores redes de sensores, capazes de fornecer dados em grande quantidade que potencialmente geram novos conteúdos e permitem o desenvolvimento de novas aplicações. Com a procura destes sistemas podemos perceber as tendências e os caminhos que têm sido percorridos nos últimos anos [26]. A Figura 2.2 representa um gráfico da distribuição das principais áreas pesquisadas. Conforme indica o gráfico, a tecnologia tem sido a área mais pesquisada, e apesar de ser um tema amplo, o foco tem sido a arquitetura do sistema e a interoperabilidade. Desta forma conseguimos perceber a necessidade da existência de interoperabilidade e que trabalhos têm sido desenvolvidos nessa área para enfrentar esse problema.

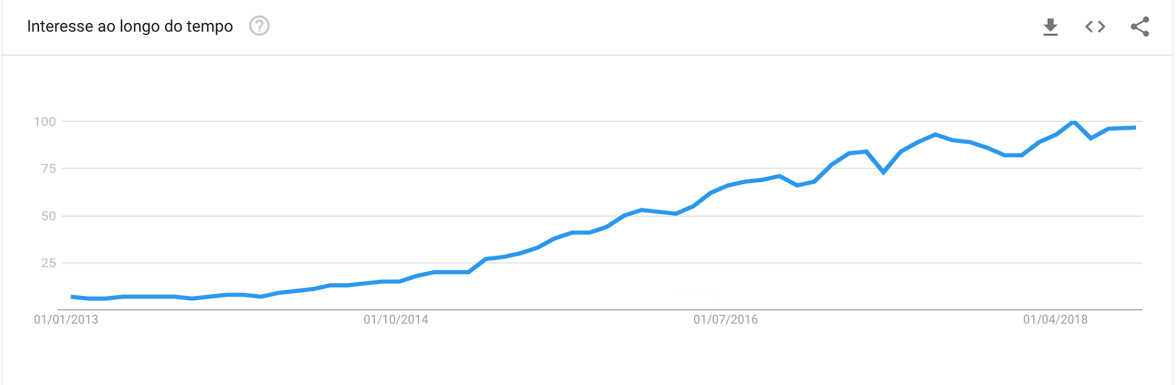

Figura 2.1: Interesse ao longo do tempo da palavra IoT[29]

Neste momento, a International Data Corporation (IDC) prevê que o mercado da IoT chegue a 1.7 biliões de dólares em quatro ou cinco anos, traduzindo-se no triplo das receitas atuais [24], sendo de extrema importância a interoperabilidade dos vários sistemas no mercado.

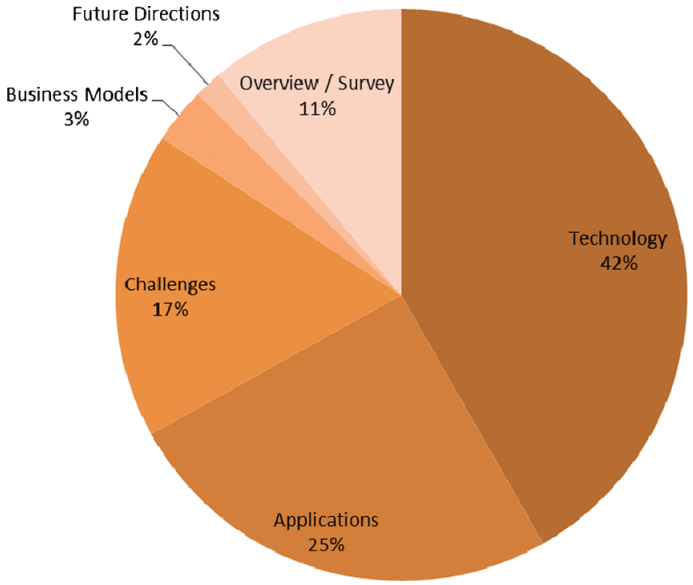

Figura 2.2: Distribuição de artigos por categoria [26]

2.6 Necessidade da IoT

Com o crescente interesse da IoT e da M2M, vem a necessidade em criar maiores redes de sensores capazes de fornecer dados de forma inteligente, afim de criar novos conteúdos e assim desenvolver novas aplicações.

Através das WSNs foi possível criar sistemas capazes de gerar dados, utilizando para tal sensores, e manipular e visualizar esses dados em aplicações. Tendo a WSN como base, criaram-se novas funcionalidades de rede, permitindo que aplicações e sensores comuniquem diretamente com os sensores utilizados na WSN. Este modelo de sistema foi inicialmente adotada devido à sua simplicidade e apelidado de sistema vertical. No entanto, este modelo não é escalável, sendo difícil a integração com outras soluções.

Afim de ultrapassar as limitações de uma solução vertical surgiu um modelo horizontal. Este modelo cria uma camada de abstração entre as aplicações e os sensores. Desta forma torna o sistema mais flexível, de fácil administração e manutenção pois os sensores estão desacoplados das aplicações. Um sistema horizontal é atualmente a melhor opção para a criação de plataformas em IoT. No entanto, a falta de especificações e padrões, para definir a estrutura da plataforma e as suas interfaces, bem como para recomendar o uso de protocolos abertos, podem comprometer a interoperabilidade entre soluções, sensores, dispositivos e aplicações.

2.7 Smart Cities

Uma de muitas aplicações da IoT são as *Smart Cities*. Estas são cidades que utilizam tecnologias de informação onde são produzidos e consumidos dados que criam novas informações, daí serem chamadas de inteligentes. Dados associados ao clima, nível das águas, condições de trânsito são algumas das muitas informações disponibilizadas como constatado na Figura 2.3. Estas informações podem ser utilizadas por programadores para a criação de aplicações referentes a uma cidade.

Os dados são, em grande parte das vezes, recolhidos a partir de sensor estrategicamente colocados para uma melhor obtenção. Esses sensores transmitem os dados recolhidos para uma plataforma centralizada que reúne todas as informações. Para tal é utilizado um *broker*.

Um *broker* serve como mediador entre um produtor e um subscritor de informação, que neste caso são sensores e aplicações.

Ao disponibilizar esse tipo de dados aos órgãos de decisão de uma cidade pretende-se motivar a criação de sistemas e plataformas que acrescentem valor aos dados recolhidos e, através disso, disponibilizar as informações e assim melhorar o quotidiano de uma cidade.

Serviços de transportes públicos podem ser melhorados e otimizados, sabendo em que alturas do dia recebem mais utilizadores e assim salvaguardar alturas de maior afluência e ajustar horários tendo em atenção situações de intensidade de trânsito na rota e outras informações dessa natureza.

A falta de interfaces padronizadas para aceder em tempo real aos dados provenientes das cidades torna-se um enorme desafio [31]. Então, aqui reside a causa de muitos dos atrasos na apresentação de soluções *Smart Cities* e IoT no geral.

Não existem padrões unânimes e utilizados por todos os fabricantes e até que estes sejam criados, a conexão de diferentes dispositivos de diferentes fabricantes geram soluções adaptadas para o efeito, não permitindo que o verdadeiro conceito de *Smart City* se torne realidade. Felizmente, existem várias cidades que começam a disponibilizar os seus dados e investem em soluções *Smart City*, porém atualmente esse esforço é feito em diferentes direções e em diferentes cidades.

Figura 2.3: Smart Cities[30]

2.8 Camadas físicas, de dados e de rede

Atualmente, muitas das tecnologias de comunicação são bem conhecidas como Wi-fi, *Bluetooth*, ZigBee e até mesmo as redes de 2G/3G/4G. Porém, existem muitas outras emergentes graças ao advento da IoT(Figura 2.4). São exemplo disso a *Sigfox* e *LoRaWAN* implementadas em algumas *Smart Cities*[32]. Dependendo da aplicação, fatores como alcance, eficiência energética, duração da bateria e segurança ditarão a escolha de uma ou alguma combinação de tecnologias. De seguida enumerar-se-ão algumas das principais tecnologias de comunicação[33].

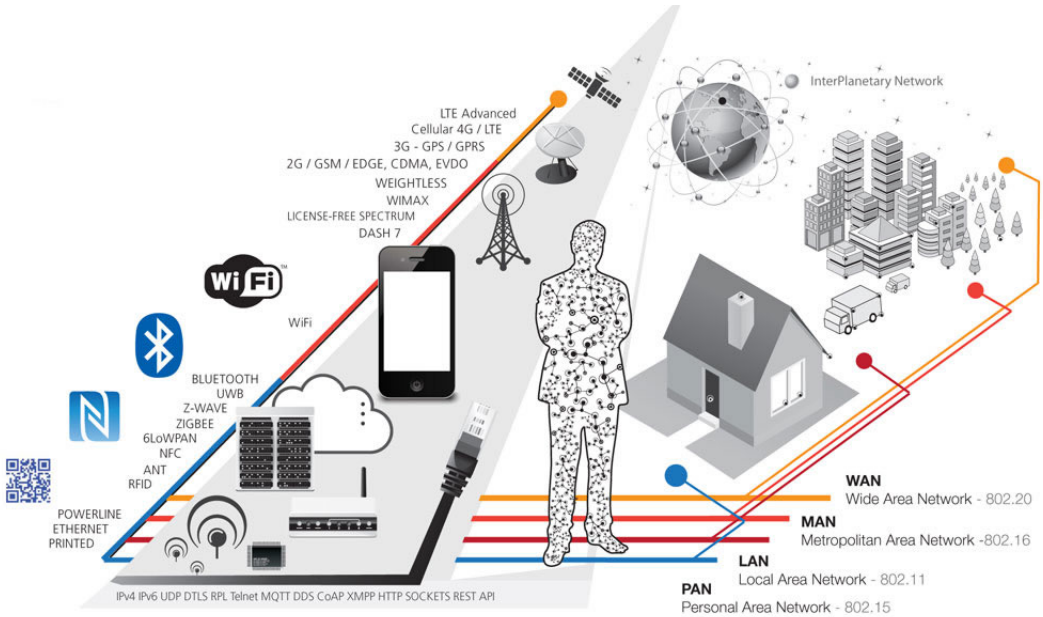

Figura 2.4: Tipologia das ligações[34]

2.8.1 NFC

A Near Field Communication (NFC) é uma tecnologia que permite a troca de informações entre dispositivos sem a necessidade de cabos, realizando as comunicações *wireless*, sendo necessária apenas uma aproximação física. Esta tecnologia teve origem no padrão Radio-Frequency IDentification (RFID), mas distanciou-se deste ao limitar o campo de atuação para uma distância máxima de 10 centímetros, tendo por objetivo tornar-se mais segura. As ligações são bidirecionais, simples e seguras entre os dispositivos, vulgarmente associadas a *smartphones*, permitindo aos utilizadores realizar pagamentos *contactless*, trocas de arquivos, entre outras operações. Para além destas funcionalidades é possível identificar funcionários e guias turísticos virtuais aproximando o *smartphone* para a obtenção de mais informações. Também é possível, com a utilização de NFC visualizar publicidade direcionada no *smartphone*, bem como recolher informações complementares de artigos expostos em superfícies comerciais.

- **Padrão:** ISO / IEC 18000-3
- **Frequência:** 13.56MHz (ISM)
- **Escala:** 10cm
- **Taxas de transferência:** 100-420kbps

2.8.2 ZigBee

O ZigBee é utilizado maioritariamente em configurações industriais. O ZigBee PRO e o Controle Remoto ZigBee (RF4CE), entre outros perfis ZigBee disponíveis, são baseados no protocolo IEEE802.15.4, que é uma tecnologia padrão de rede sem fio, utilizando os 2,4 GHz. O alcance é relativamente reduzido, permitindo interligar dispositivos dentro de uma casa ou edifício.

O ZigBee/RF4CE é utilizado em dispositivos de baixa potência, robustos e com alta escalabilidade o que o torna um bom candidato para ser utilizado em soluções M2M e IoT.

- **Padrão:** ZigBee 3.0
- **Frequência:** 2.4GHz
- **Escala:** 10-100m
- **Taxas de transferência:** 250kbps

2.8.3 Z-Wave

A *Z-Wave* é uma tecnologia de comunicações Radiofrequência (RF) de baixa potência, projetado para automação residencial, tipicamente utilizado em controladores de lâmpadas e sensores. Está otimizado para comunicações confiáveis de pequenos pacotes de dados com taxas de transferência de dados até 100kbps e baixa latência. Permite a criação de redes *mesh* sem a necessidade de um nó central e é bastante escalável, permitindo o controlo de até 232 dispositivos. O *Z-Wave* permite um desenvolvimento rápido e simples.

- **Padrão:** *Z-Wave* Alliance ZAD12837 / UIT-T G.9959
- **Frequência:** 900MHz (ISM)
- **Escala:** 30m
- **Taxas de transferência:** 9,6/40/100kbps

2.8.4 Bluetooth

O *Bluetooth* é uma das tecnologias de comunicação de curto alcance mais utilizada atualmente. O Bluetooth Low-Energy (BLE) é um protocolo com uma forte aplicação em soluções IoT. Apesar de utilizar uma gama de frequência idêntica ao *Bluetooth*, o BLE foi projetado para permitir um consumo de energia significativamente mais reduzido.

No entanto, o BLE não foi projetado para transferência de ficheiros, mas sim para comunicar pequenas quantidades de dados. Atualmente a maioria dos smartphones já vem equipada com esta tecnologia, o que permite uma integração simples com outros dispositivos.

- **Padrão:** *Bluetooth* 5
- **Frequência:** 2.4GHz (ISM)
- **Escala:** 50-150m (BLE)
- **Taxas de transferência:** 1Mbps (BLE)

2.8.5 6LowPAN

IPv6 over Low power Wireless Personal Area Networks (6LowPAN) é um padrão da IETF, entidade responsável por muitos dos padrões utilizados na *Internet*, como User Datagram Protocol (UDP), Transmission Control Protocol (TCP) e HyperText Transfer Protocol (HTTP). Com este padrão é pretendido adaptar os pacotes do protocolo Internet Protocol version 6 (IPv6) ao ambiente de redes pessoais de baixa potência, como as definidas pelo padrão IEEE 802.15.4 e também conhecidas como redes de sensores sem fio 6LowPAN. Permite que os dispositivos mais pequenos e/ou com capacidade de processamento limitada transmitam informações através de ligações sem fio usando um protocolo de *Internet*.

A IETF tem proposto nos últimos anos algumas Request for Comments (RFC) sobre adaptações de uso da nova versão do protocolo de *Internet* IPv6 para as redes Low power Wireless Personal Area Networks (LowPAN). O principal objetivo é adaptar os pacotes IPv6 para que estes possam ser utilizados em redes LowPAN e vice-versa, integrando essas redes nas redes IP convencionais e, por consequência, à *internet*. Essa integração permite uma nova variedade de aplicações como monitorização de ambientes industriais, agricultura, meio ambiente, infraestrutura de construções, monitorização da saúde de pacientes, telemetria, *smart cities*, entre outras [35].

- **Padrão:** RFC6282
- **Frequência:** *Bluetooth Smart* (2.4GHz) ou ZigBee ou RF de baixa potência (sub-1GHz)
- **Escala:** N / A
- **Taxas de transferência:** N / A

2.8.6 Wifi

A utilização de *Wifi* é muitas vezes uma escolha óbvia para o desenvolvimento de muitos projetos, devido especialmente à difusão do *Wifi* em ambientes domésticos dentro de uma Local Area Network (LAN). O *Wifi* tem uma grande utilização graças a uma ampla infraestrutura já existente, além de oferecer uma transmissão rápida e capacidade para lidar com grandes quantidades de dados.

Atualmente, o padrão *Wifi* mais utilizado em casas e muitas empresas é o 802.11n, pois oferece uma elevada taxa de transferência de dados, sendo apropriado para a transferências de arquivos. Mesmo não tendo sido criado com esse propósito, a utilização do *Wifi* tem sido muito útil em aplicações de IoT, graças às razões apresentadas anteriormente.

- **Padrão:** 802.11n
- **Frequência:** bandas de 2,4GHz e 5GHz
- **Escala:** Aproximadamente 50m
- **Taxas de transferência:** 150-200Mbps (típico), 500Mbps-1Gbps (802.11-ac)

2.8.7 LoRaWAN

LoRaWAN é um protocolo de controlo de acesso (Mandatory Access Control (MAC)) para Wide Area Network (WAN). Foi projetado para permitir que dispositivos de baixa potência e recursos limitados comuniquem com aplicações ligadas à *internet* através de ligações sem fio de longo alcance. Suporta comunicações bidirecionais de baixo custo em IoT, M2M, *Smart Cities* e aplicações industriais. *LoRaWAN* pode-se fazer corresponder à segunda e terceira camada do modelo Open System Interconnection (OSI).

O protocolo partilha alguns aspetos com *Sigfox* e *Neul* e foi otimizado para consumo de baixa potência, suporte de grandes redes de dispositivos e as taxas de transferência variam de 0,3kbps a 50kbps.

- **Padrão:** LoRaWAN
- **Frequência:** Várias
- **Escala:** 2-5km (ambiente urbano), 15Km (ambiente suburbano)
- **Taxas de transferência:** 0,3-50kbps

2.8.8 Sigfox

A *Sigfox* é uma multinacional francesa, operadora de telecomunicações, responsável por desenvolver redes sem fios que ligam equipamentos de baixo consumo, como por exemplo contadores de eletricidade, água e gás, eletrodomésticos, iluminação pública e sensores ambientais à *Internet*.

A tecnologia *Sigfox* utiliza as bandas Industrial, Scientific and Medical (ISM) gratuitas, sem a necessidade da aquisição de licenças para transmitir dados entre dispositivos.

As redes criadas usando o protocolo *Sigfox* são denominadas de Low-Power Wide-Area Network (LPWAN). Estas redes são especialmente concebidas para permitir comunicações de longo alcance. A sua velocidade de comunicação é geralmente bastante reduzida (10-1000 bps) devido ao seu elevado comprimento de onda e reduzida largura de banda. O aparecimento destas redes veio permitir a expansão das comunicações M2M, já que permitiu baixar o consumo energético dos dispositivos ligados e reduzir o preço dos dispositivos.

Para obter uma solução que implemente a *stack Sigfox* é necessário possuir um módulo de comunicação que pode ser fornecido por vários fabricantes (e sempre validado pela empresa) e cobertura na área geográfica onde o sistema vai ser implementado, que pode ser obtida no

web-site da empresa. No primeiro trimestre de 2018, a *Sigfox* atingiu uma área de 17 milhões de km^2 e uma população superior a 465 milhões de pessoas[36].

A rede é robusta, eficiente em termos energéticos, escalável e pode interligar milhões de dispositivos em áreas de vários quilómetros quadrados. Esta rede é portanto adequada para várias aplicações M2M que podem incluir contadores inteligentes, iluminação pública e sensores ambientais.

O *Sigfox* utiliza tecnologia Ultra Narrow Band (UNB) para estabelecer uma comunicação bidirecional entre equipamentos e uma estação base proprietária. Assim que uma estação recebe uma mensagem dum dos dispositivos na área que esta cobre, envia a mensagem para a *cloud* da Sigfox. Depois deste processo, a informação é enviada para o utilizador, onde este consegue visualizar e manipular a informação obtida.

A tecnologia UNB possibilita o envio de mensagens em canais com largura de banda geralmente inferiores a 200Hz. O alcance deste tipo de sinais pode superar os 30km em ambientes suburbanos. Outra das vantagens da tecnologia UNB é a forte imunidade a ruído: como o sinal tem uma largura de banda muito pequena, os recetores terão filtros também com uma largura de banda pequena, removendo assim uma grande parte do ruído do sinal.

- **Padrão:** Sigfox
- **Frequência:** 900MHz (ISM)
- **Escala:** 3-10Km (ambientes urbanos), 30-50Km (ambientes suburbanos)
- **Taxas de transferência:** 10-1000bps

2.8.9 Neul

Tal e qual o protocolo *Sigfox* que opera na banda 900MHz, o *Neul* utiliza parte do espectro para a criação de redes de alto escalabilidade, alta cobertura, baixa potência e baixo custo. A tecnologia de comunicação é chamada de *Weightless*, uma nova tecnologia de redes sem fios WAN projetada para soluções IoT que pretende ser uma alternativa para soluções existentes de GPRS, 3G, CDMA e WAN LTE. As taxas de transferência de dados pode atingir um máximo de 100kbps e os dispositivos podem consumir apenas 20 a 30mA.

- **Padrão:** Neul
- **Frequência:** 900MHz (ISM), 458MHz (UK), 470-790MHz (White space)
- **Escala:** 10Km
- **Taxas de transferência:** 100kbps (máximo)

2.8.10 Mobile

Qualquer aplicação IoT que necessite de efetuar comunicações em longas distâncias pode aproveitar as capacidades de comunicação GSM/3G/4G. Embora as comunicações mobile sejam, claramente, capazes de enviar grandes quantidades de dados, especialmente em 4G, o consumo de energia é extremamente elevado para muitas aplicações, mas pode ser ideal para projetos com baixa largura de banda baseados na recolha de dados dos sensores e enviam grandes quantidades de dados através da *internet*[33].

- **Padrão:** GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)

- **Frequência:** 900/1800/1900/2100MHz
- **Escala:** 35Km máximo para GSM
- **Taxas de transferência:** 35-170kps (GPRS), 120-384kbps (EDGE), 384Kbps-2Mbps (UMTS), 600kbps-10Mbps (HSPA), 3-10Mbps (LTE)

2.8.11 Secção de Comparação

A NFC é uma tecnologia sem fios que permite a transmissão de dados através de comunicações bidirecionais numa distância máxima de 10 centímetros. A taxa de transferência é de 100-420kbps.

O *ZigBee* utiliza a frequência de 2,4GHz nas suas comunicações. O alcance do sinal fica pelos 10-100m utilizando para tal uma taxa de transferência de 250kbps.

A *Z-Wave* é uma tecnologia de comunicações RF de baixa potência. Com taxas de transferência de dados até 100kbps e baixa latência, permitindo um alcance de 30m e o controlo de até 232 dispositivos.

O *Bluetooth* é uma das tecnologias de comunicação de curto alcance mais utilizada atualmente. O BLE foi projetado para permitir um consumo de energia significativamente mais reduzido com um alcance de 50m-150m e uma taxa de transferência de 1Mbps.

6LowPAN é um padrão que pretende adaptar os pacotes do protocolo IPv6 ao ambiente de redes pessoais de baixa potência, como as definidas pelo padrão IEEE 802.15.4.

O padrão *Wifi* oferece uma elevada taxa de transferência de dados tipicamente de 150-200Mbps com um alcance máximo de 50m.

O protocolo *LoRaWAN* partilha alguns aspetos com *Sigfox* e *Neul* e foi otimizado para consumo de baixa potência com taxas de transferência de 0,3kbps-50kbps e um alcance de 2km-15km dependente do ambiente envolvente.

A tecnologia *Sigfox* utiliza as bandas ISM gratuitas, sem a necessidade da aquisição de licenças para transmitir dados entre dispositivos. Estas redes são especialmente concebidas para permitir comunicações de longo alcance com uma taxa de transferência de 10-1000bps. O alcance deste tipo de sinais pode superar os 30km em meios suburbanos.

Tal e qual o protocolo *Sigfox*, o protocolo *Neul* que opera na banda dos 900MHz tem taxas de transferência de dados que podem atingir os 100kbps com um alcance de cerca de 10km.

Aplicações IoT que utilizem comunicações GSM/3G/4G conseguem ter um alcance de cerca de 35km, sem que seja necessário a repetição do sinal. Por outro lado conseguem taxas de transferência que vão desde 35-170kps a 3-10Mbps dependendo a tecnologia utilizada.

2.9 Camadas de transporte e aplicação

A IoT é um termo usado para descrever uma rede que permite a interligação de computadores, sensores, atuadores e dispositivos moveis para que comuniquem entre si, seja através de ligações com e/ou sem fio [37]. Assim, estes dispositivos permitem ler os diversos sensores e tomar decisões, muitas vezes sem a intervenção humana. Isto é obtido através do uso de telemetria que permite a comunicação dos vários dispositivos. Tal comunicação foi originalmente obtida através de uma rede de dispositivos que através da transmissão de dados para um agregador,

analisava e posteriormente encaminhava essas informações para outros dispositivos [38]. Para que essas comunicações sejam possíveis existem vários protocolos de IoT que serão apresentados de seguida.

2.9.1 HTTP

O HTTP é um protocolo de comunicação de dados para a World Wide Web (WWW). O desenvolvimento do HTTP foi coordenado pela IETF e pelo World Wide Web Consortium (W3C) e culminou na criação da RFC2616 onde foi publicada o HTTP/1.1.

HTTP tem por base o modelo cliente-servidor onde o cliente solicita ao servidor um determinado conteúdo HTTP. Após essa solicitação o servidor executa algumas ações e em nome do cliente que as solicitou retorna uma mensagem de resposta. O código da resposta indica que o pedido foi concluído e este poderá conter algum conteúdo no corpo da mensagem.

Este protocolo foi criado para permitir comunicações entre clientes e servidores e é também utilizado em soluções IoT. Isto acontece pela utilização do modelo Representational State Transfer (REST) muito utilizado em sistemas que possuem um servidor central que recebe pedidos de um ou mais clientes, possibilitando que o processamento seja mantido do lado do servidor, reduzindo assim a carga de processamento dos clientes.

Em IoT são utilizados vários serviços REST baseados em HTTP para comunicação com dispositivos remotos [39].

Num caso de uso em que um dispositivo munido de sensores recolhe dados e os envia para um servidor central, o diapositivo poderá fazer um pedido HTTP POST enviando os dados no corpo do mesmo.

2.9.2 MQTT

O Message Queuing Telemetry Transport (MQTT) é um protocolo leve de mensagens *publish/subscribe* executado sobre o protocolo TCP/IP. As suas características tornam-o a escolha ideal para implementações de M2M em contexto de IoT.

MQTT usa um padrão de mensagem *publish/subscribe* que permite uma distribuição de mensagens de um para muitos e um cliente pode subscrever vários canais. Estes canais podem ser subscritos a qualquer momento, bem como o cancelamento dos mesmos. Uma das maiores vantagens é o seu baixo *overhead*.

São suportados três tipos de Quality of Service (QoS) na entrega das mensagens:

- QoS 0 - *"At most once"*

  O nível mínimo de QoS é 0 e as mensagens são entregues no máximo uma vez, ou não é entregue de todo. As mensagens não são armazenadas, sendo possível perder uma mensagem caso o recetor esteja desligado ou se o servidor falhar. O nível de QoS 0 é o modo de transferência mais rápido, porém o recetor não se importa caso algumas mensagens sejam perdidas.
- QoS 1 - *"At least once"*

  Ao usar QoS nível 1, é garantido que uma mensagem será entregue pelo menos uma vez ao recetor e esta poderá ser entregue mais de uma vez. QoS nível 1 é o modo padrão

de transferência. Se o remetente não receber uma confirmação através de um PUBACK, a mensagem será enviada novamente com o sinalizador DUP marcado até que uma confirmação seja recebida. A mensagem é excluída do recetor após ser processada a mensagem. Se o destinatário for um *broker*, a mensagem é recebida, armazenada e de seguida publicada nos seus *subscribers*. A mensagem só é excluída após o remetente receber a confirmação de receção por parte do destinatário.

- QoS 2 - *"Exactly once"*

  Utilizando QoS 2, as mensagens são sempre entregues exatamente uma vez. A mensagem deve ser armazenada localmente no remetente e no recetor até que seja processada. QoS nível 2 é o modo de transferência mais seguro, porém a mais lenta. É preciso pelo menos dois pares de transmissões entre o remetente e o recetor antes que a mensagem seja excluída do remetente. A mensagem pode ser processada no recetor após a primeira transmissão.

  No primeiro par de transmissões, o remetente transmite a mensagem e recebe a resposta PUBREC do recetor que armazenou a mensagem. Se o remetente não receber uma confirmação, a mensagem será enviada novamente com o sinalizador DUP até que uma confirmação seja recebida. No segundo par de transmissões, o remetente informa ao recetor que pode completar o processamento da mensagem através de PUBREL. Se o remetente não receber uma confirmação da mensagem PUBREL, a mensagem PUBREL será enviada novamente até receber uma confirmação. O remetente exclui a mensagem que guardou quando recebe a confirmação da mensagem PUBREL.

2.9.3 CoAP

O protocolo CoAP permite que dispositivos comuniquem através da *internet*, especialmente sensores de baixa potência que requerem administração remota, tornando-o num protocolo muito útil para cenários de *Smart Cities* e inevitavelmente para IoT.

O propósito do surgimento do CoAP foi a criação de uma aproximação ao HTTP para assim permitir uma integração mais fácil com a *web*, e ao mesmo tempo suportar requisitos de *multicast* mantendo a simplicidade. Esses requisitos são de extrema importância para a IoT porque os dispositivos que dão uso ao protocolo têm recursos muito limitados. Este protocolo pode funcionar na maioria dos dispositivos que suportam UDP [40]. A utilização do CoAP é semelhante à utilização do HTTP como a consulta de uma Application Programming Interfaces (APIs) na *internet*.

Além disso, existe uma extensão do protocolo que permite aos clientes CoAP conhecer recursos, as suas representações e mantê-los atualizados ao longo do tempo.

Alguns resultados comparando CoAP e HTTP indicam que existe uma grande variedade de situações onde o uso do CoAP é mais eficiente em relação ao HTTP[40]. Exemplo disso é quando um dispositivo se mantém adormecido entre períodos de transmissão dos dados. O CoAP é especialmente benéfico em relação ao cenários onde os dispositivos se encontrem ativos na maior parte do tempo. O CoAP é também bastante amigável graças ao seu baixo *overhead* a quando da transmissão dos dados.

2.9.4 XMPP

O eXtensible Messaging and Presence Protocol (XMPP) é um protocolo baseado em eXtensible Markup Language (XML), projetado originalmente para a transmissão de mensagens instantâneas e deteção de presenças online. Existe uma extensão que especifica como os dispositivos podem ser instalados e descobertos em segurança em redes IoT, para tal fornece um modo de instalação, configuração, pesquisa e ligação.

Se por um lado a instalação de dispositivos IoT em redes públicas deve ser simples, por outro deve ser também seguro. Por estas razões esta tecnologia é um potencial candidato a ser utilizado em cenários IoT. Existem muitos casos de uso para a aplicação desta tecnologia num cenário IoT, mas os principais são produção, instalação, localização de um servidor XMPP e sua conexão. A produção é a fase de atribuição dos parâmetros *default* para a configuração e ligação à rede. A instalação é o momento do processo onde são configurados alguns parâmetros extra num ambiente de produção. A descoberta de um servidor XMPP pode ser configurado por via manual ou através de Dynamic Host Configuration Protocol (DHCP). Por fim, e após todas as configurações e ligação a um servidor XMPP, o dispositivo fica disponível na rede.

Como estas funcionalidades associadas a IoT ainda estão em fase experimental no protocolo *XMPP-IoT*, alguns dos recursos de segurança ainda precisam ser decididos e implementados e o uso desta especificação para um ambiente de produção ainda não é recomendado.

2.9.5 LwM2M

Lightweight Machine to Machine (LwM2M) é um protocolo definido pela Open Mobile Alliance (OMA) que pretende gerir dispositivos em redes M2M e IoT.

Este protocolo define as comunicações na camada de aplicação do modelo OSI entre um servidor e um cliente. Tal como acontece com outros protocolos, os dispositivos utilizados são limitados em capacidade de processamento e armazenamento e por esse motivo este protocolo é utilizado frequentemente em conjugação com o protocolo CoAP.

O LwM2M fornece funcionalidades de gestão de dispositivos, transferência de dados do servidor para os clientes e é possível estender as funcionalidades base consoante os requisitos pretendidos.

2.9.6 Secção de Comparação

O HTTP é um dos protocolos mais utilizados em cenários web, porém protocolos como CoAP e MQTT foram projetados para serem utilizados em cenários onde os dispositivos têm menor capacidade de processamento e lidam com volumes de dados muito inferiores.

Uma das grandes vantagens do MQTT é a existência de vários níveis de QoS, estas evitam a perda de pacotes e é uma característica importante para certas aplicações em *Smart City*.

Tal e qual o protocolo HTTP, o CoAP foi construído tendo por base uma abordagem *RESTful*. Assim possibilita a disponibilização dos dados de dispositivos na *web* de uma maneira simples.

O protocolo *XMPP-IoT*, apesar de ainda não ser uma solução recomendada dado o seu estado experimental, tem mostrado algum potencial em cenários IoT.

Por fim, o LwM2M é um protocolo muitas vezes utilizado com o protocolo CoAP em cenários M2M e pretende ser eficiente e simples. A vantagem da sua implementação com o protocolo CoAP é a fácil gestão dos vários dispositivos.

2.10 Interoperabilidade

Uma verdadeira *Smart City* só poderá ser alcançada quando as diferentes partes da solução atingirem um estado de maturação que permita a interoperabilidade. Por isso é de extrema importância que as diferentes partes de uma arquitetura sejam capazes de entender e comunicar entre si [31].

Existe atualmente a necessidade de criar uma especificação única que permita a intercomunicação entre os diversos dispositivos IoT. Esta necessidade vem no seguimento da existência de inúmeros fabricantes de dispositivos e sensores que necessitam de entender os diversos mecanismos e padrões, sem esse esforço não será possível atingir a interoperabilidade.

Todos fabricantes poderiam simplesmente concordar com uma especificação única e usá-la exclusivamente, no entanto, não é assim tão simples, pois diferentes fabricantes defendem diferentes especificações. Além disso, os grandes fabricantes estão sempre a tentar que as suas especificações, que por vezes são proprietárias, sejam adotadas pela comunidade IoT.

O objetivo principal desta dissertação é fornecer uma alternativa à adoção tradicional de uma tecnologia e chegar a um ponto em que a plataforma é compatível com mais do que um padrão para alcançar uma verdadeira interoperabilidade entre eles.

2.11 Plataformas IoT

A evolução das soluções de *middleware* em M2M criou a necessidade para que as plataformas de *software* lidassem com as comunicações bem como a criação de serviços/aplicações, integrando tudo no contexto da IoT. Portanto, novos modelos de negócios começaram a surgir, permitindo às empresas criar novas soluções para cenários IoT, tanto fornecendo dispositivos como desenvolvendo aplicações.

No âmbito da saúde têm surgido várias aproximações académicas como é o caso da *INTER-IoT* [41] que permite suportar aplicações interdisciplinares e tarefas de processamento especializado. Esta aplicação permite a partilha de dados em larga escala e colaboração dos vários utilizadores utilizando para tal a *cloud* para a partilha de serviços de apoio à decisão baseada em dados provenientes de uma BSN. Esta plataforma é subdividida em quatro componentes: *Body*, *Cloud*, *Viewer* e *Analyst*. O *Body* é o componente baseado em *SPINE*[42] que permite monitorizar um paciente através de sensores portáteis e armazena os dados na *cloud* através de um dispositivo móvel. O componente *Viewer* permite, através de *browser*, visualizar os dados através de vários relatórios. Por fim o *Analyst* é o componente responsável pela criação dos relatórios e através deste é possível extrair informações a partir dos dados.

À semelhança do *INTER-IoT* existem outros projetos que assentam no mesmo funcionamento como é o caso do CodeBlue [43], DexterNet [44], SPINE2 [45], The Missouri S&T [46].

Ao nível de aplicações comerciais existem várias opções no mercado como *ThingWorx IoT Platform* da PTC, *Kaa IoT Platform* da Kaa, *The Intel IoT Platform* da Intel, *thethings.iO*, *IBM Watson Internet of Things Platform* da IBM entre muitas outras.

Neste momento, existem dezenas de plataformas IoT, tanto *open source* como comerciais. Esta seção dará uma breve visão geral sobre algumas dessas soluções.

2.11.1 ThingWorx IoT Platform

A plataforma *ThingWorx* cria condições para que empresas consigam impulsionar e inovar nos seus negócios. Para tal a plataforma fornece funcionalidades, flexibilidade e escalabilidade para sintetizar dados e orquestrar processos através de uma interface web.

Através do *ThingWorx* é possível, por exemplo, monitorizar remotamente as condições de um paciente. Os principais fabricantes de dispositivos médicos têm vindo a adotar a IoT para apoiar a transição para um serviço de "cuidados responsáveis".

Assim é possível melhorar a experiência do cliente, através de representações dos dados provenientes dos diversos dispositivos. Será também possível criar e otimizar fluxos e modelos de negócio. Ao combinar dados em tempo real com sistemas já existentes é possível aumentar a eficiência e assim diferenciar a oferta de produtos e serviços, potenciando o crescente ritmo de inovação.

Ao alavancar estas oportunidades as empresas podem encontrar novas formas de inovar e crescer.

2.11.2 Kaa IoT Platform

Kaa é uma plataforma *middleware* com diversas aplicações em IoT que permite a construção de soluções completas de P2P interligando aplicações e dispositivos. A plataforma *Kaa* (Figura 2.5) fornece um conjunto de ferramentas que permitem o desenvolvimento de soluções IoT e, portanto, reduz drasticamente o custo, os riscos e o tempo de desenvolvimento.

Existem uma conjunto de detalhes na arquitetura desta plataforma que tornam o desenvolvimento com *Kaa* rápido e fácil em soluções IoT. *Kaa* é agnóstica ao *hardware* utilizado e, portanto, compatível com praticamente qualquer tipo de dispositivo, sensor e/ou *gateway*.

Esta plataforma fornece uma estrutura clara dos recursos e extensões de IoT para diferentes tipos de aplicações. Podem ser usados quase como módulos Plug and Play (PnP) com a utilização mínima de código adicional. Por outro lado permite a utilização de diversos protocolos de comunicação. Estas funcionalidades fazem de *Kaa* uma plataforma adequada para o desenvolvimento de soluções para IoT. Kaa é *open-source*.

Figura 2.5: Estrutura da plataforma Kaa IoT[47]

2.11.3 The Intel IoT Platform

A Intel com a criação da sua plataforma para cenários IoT pretendeu acompanhar o crescente movimento para a interligação de dispositivos inteligentes e sensores, e assim realizar análises de grandes quantidades de dados.

A *Intel IoT Platform* é um modelo de referência de P2P e tem vindo a trabalhar em soluções de terceiros para interligar dispositivos de forma transparente e segura. Os dados são recolhidos e através de uma solução na *cloud* são tratados e analisados, utilizando para tal comunicações confiáveis.

A plataforma permite reduzir a complexidade da solução não descurando a segurança, interoperabilidade, escalabilidade e gestão.

Para tal, a Intel através de parcerias, tem adicionado *software* e serviços para que mais dispositivos se interliguem à sua plataforma. A Intel descreveu a sua plataforma como "um modelo de referência de P2P projetada para unificar e simplificar as comunicações e segurança na IoT".

2.11.4 thethings.io

thethings.io é uma plataforma IoT que permite uma conexão rápida e escalável dos vários dispositivos à *internet*, permitindo a monitorização e administração dos dispositivos em tempo real e obtenção de relatórios analíticos flexíveis. Para tal, esta plataforma fornece uma solução de *backend* completa para interligação de dispositivos IoT e manuseamento dos dados provenientes dos diversos dispositivos através de uma API fácil e flexível.

Esta plataforma é agnóstica ao *hardware* utilizado, isso significa que não requer *hardware* específico, bastando que esse *hardware* seja capaz de suportar qualquer um dos protocolos: HTTP, *Websockets*, MQTT ou CoAP.

2.11.5 IBM Watson Internet of Things Platform

A *Watson Internet of Things Platform* é um serviço alojado e gerido a partir da *cloud*. Utiliza uma solução de Platform as a Service (PaaS), o que permite em qualquer momento adicionar e gerir os dispositivos.

Os dispositivos são adicionados à plataforma *Bluemix* e poderão ser sensores, *gateway* ou qualquer outro dispositivo. Através de comunicações seguras é possível a troca de dados em tempo real.

As comunicações são realizadas utilizando o protocolo de mensagens MQTT. A partir daí, é possível configurar os vários dispositivos utilizando um *frontend* ou uma API disponível para o efeito, que permite visualizar dados históricos bem como dados em tempo real.

2.11.6 Meshblu

O *Meshblu* permite a troca de mensagens instantâneas em redes M2M. Fornece uma API que está disponível através de REST em HTTP, Web Sockets, MQTT e CoAP. A principal função é fazer a ponte entre dispositivos que suportem protocolos diferentes. Assim sendo será possível que dispositivos que apenas suportam CoAP consigam comunicar com dispositivos que apenas suportem MQTT.

Inicialmente os dispositivos são registados e é-lhes atribuído um Universal Unique Identifier (UUID) e respetivos *tokens* para utilização em futuras comunicações entre dispositivos.

Através do *Meshblu* é possível descobrir, consultar e enviar mensagens entre dispositivos. Para além destas funcionalidades é possível subscrever mensagens e atividades dos vários sensores.

2.11.7 Ponte by Eclipse

Através de *Ponte* é possível enviar e receber dados através de HTTP, MQTT ou CoAP. É possível inclusive enviar os dados em formatos diferentes.

Os dispositivos podem obter atualizações em tempo real através de MQTT ou CoAP, bastando para tal subscrever respetivamente os métodos correspondentes. Depois disso, os utilizadores podem receber notificações em tempo real através de *MQTT-over-Websockets*.

É possível utilizar vários formatos de dados sejam eles JavaScript Object Notation (JSON), Message PackData Stream Management Systems (MsgPack) e XML.

Tal e qual acontece com *Meshblu*, o *Ponte* permite autenticar e configurar os dispositivos.

Através de *Ponte* é possível criar aplicações baseadas em REST aproximando-as a cenários de IoT. Assim é possível que protocolos IoT interajam com protocolos *web* e vice-versa.

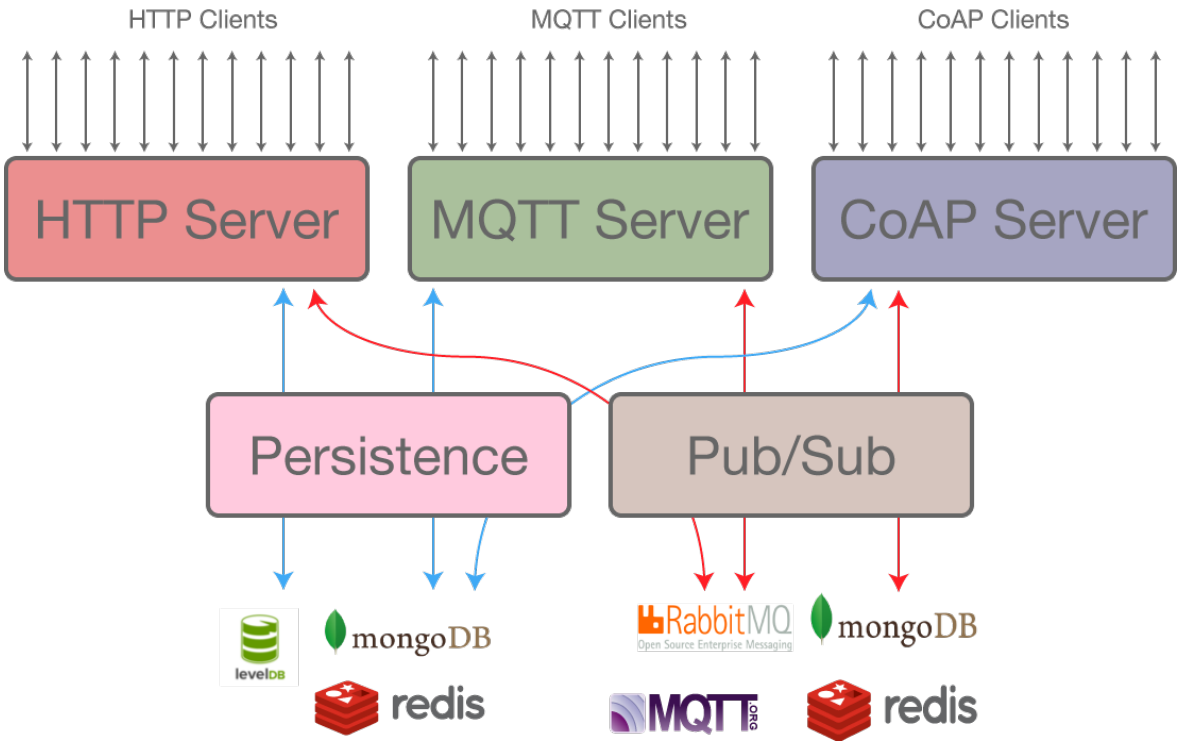

Figura 2.6: Arquitetura Ponte by Eclipse[48]

2.11.8 hipercat

O *Hypercat* é um consórcio e padrão que visa impulsionar uma IoT segura e verdadeiramente interoperável. As especificações permitem aos dispositivos descobrir quais os recursos disponíveis num servidor IoT. A implementação foi baseada em padrões bastante conhecidos, como HyperText Transfer Protocol Secure (HTTPS), REST e JSON.

O consórcio inicialmente foi composto por cerca de quarenta empresas, atualmente são mais de 70 empresas e o principal objetivo foi a procura de semelhanças entre as arquiteturas já existentes.

Inicialmente começaram por analisar as interfaces dos níveis inferiores e descobriram que existem imensos protocolos diferentes para as interligação dos dispositivos. Rapidamente se percebeu que alcançar a interoperabilidade seria uma tarefa deveras difícil.

Em vez de procurarem um solução de baixo nível, o foco foi para as camadas superiores. Perceberam rapidamente que a maioria das plataformas utilizam padrões *web* conhecidos como HTTPS, REST e JSON. O objetivo principal passou pela criação de aplicações que lidassem com diferentes fontes de dados.

Criaram interfaces comuns para as várias aplicações, porém tiveram de lidar com novos problemas. A maneira como as informações são exibidas e organizadas difere dependendo da origem dos dados. Os dados são transmitidos e fazem sentido apenas para a fonte desses mesmos dados, mas não necessariamente para todas as outras. A "semântica" varia de fonte para fonte e o mesmo recurso não é chamado sempre da mesma maneira.

Atualmente um programador tem de conhecer a documentação bem como a organização

dos dados afim de criar aplicações de acordo com os dados provenientes dos dispositivos. Sem dúvida que esse é um dos princípios problemas da IoT, pois assim o programador será obrigado a criar *software* ajustado a cada um das fontes de dados, já para não falar que uma solução deste tipo não é escalável. À medida que o número de aplicações cresce, cresce também os dados gerados e dependendo do cenário é necessário desenvolver software especifico para cada um. É claro que isso não é viável.

A solução mais simples passaria pela uniformização e organização da semântica utilizada nas diversas APIs. Isso foi considerado uma abordagem inviável porque existem razões para essas mesmas diferenças.

Sabendo isso, o consórcio *Hypercat* rapidamente decidiu que o caminho a seguir passaria pela criação de uma maneira de as máquinas resolverem automaticamente o problema, sem seres humanos, interpretando e interligando dados com organização e semântica diferente. Com estes princípios surgiu o *Hypercat*.

2.11.9 OpenMTC

A plataforma Open Machine Type Communications (OpenMTC) implementa um *middleware* M2M à semelhança de outros padrões: *oneM2M*, OMA LwM2M, European Telecommunications Standards Institute (ETSI) M2M e 3rd Generation Partnership Project (3GPP). O OpenMTC é utilizado para interligar vários sensores e atuadores diferentes. É uma plataforma na *cloud* que transmite os dados a partir dos dispositivos para as diversas aplicações. A interação é realizada a partir de eventos despoletados a partir dos dispositivos.

É composto por dois Service Capability Layers (SCL), um Gateway Service Capability Layers (GSCL) e um Network Service Capability Layers (NSCL). O GSCL suporta múltiplas tecnologias M2M e protocolos de comunicação, enquanto a NSCL é uma plataforma M2M baseada na *cloud* para agregação e armazenamento dos dados.

Através do IP Multimedia Subsystem (IMS) e Evolved Packet Core (EPC) é possível a interação com a infraestrutura de telecomunicações. Assim é possível trocar informações com os sensores dos dispositivos através de uma comunicação Session Initiation Protocol (SIP).

Assim, potencialmente as aplicações M2M têm uma maior nível de confiança nas implementações existentes do IMS, bem como nas capacidades de QoS do EPC.

Para facilitar e melhorar o desenvolvimento de aplicações M2M, a plataforma OpenMTC fornece um Software development kit (SDK), disponibilizando recursos básicos para aplicações de terceiros. A Figura 2.7 mostra uma representação geral da arquitetura da plataforma OpenMTC, dando uma melhor compreensão sobre como tudo se encaixa.

A OpenMTC otimiza a rede integrando elementos 3GPP para obter o estado da ligação sem a utilização de um canal dedicado para o efeito. Esta plataforma pode ser implementada em dispositivos com recursos limitados bem como dispositivos compatíveis com Linux e Android, enquanto que o *backend* pode ser implementado na *cloud*, permitindo escalar a plataforma. Por outro lado, com a utilização do formato Open XML Paper Specification (OpenXPS), é possível manipular dados com origem nos sensores e torná-los legíveis, convergindo as comunicações M2M e Human-to-Machine (H2M). OpenMTC oferece protocolos multi-transporte como

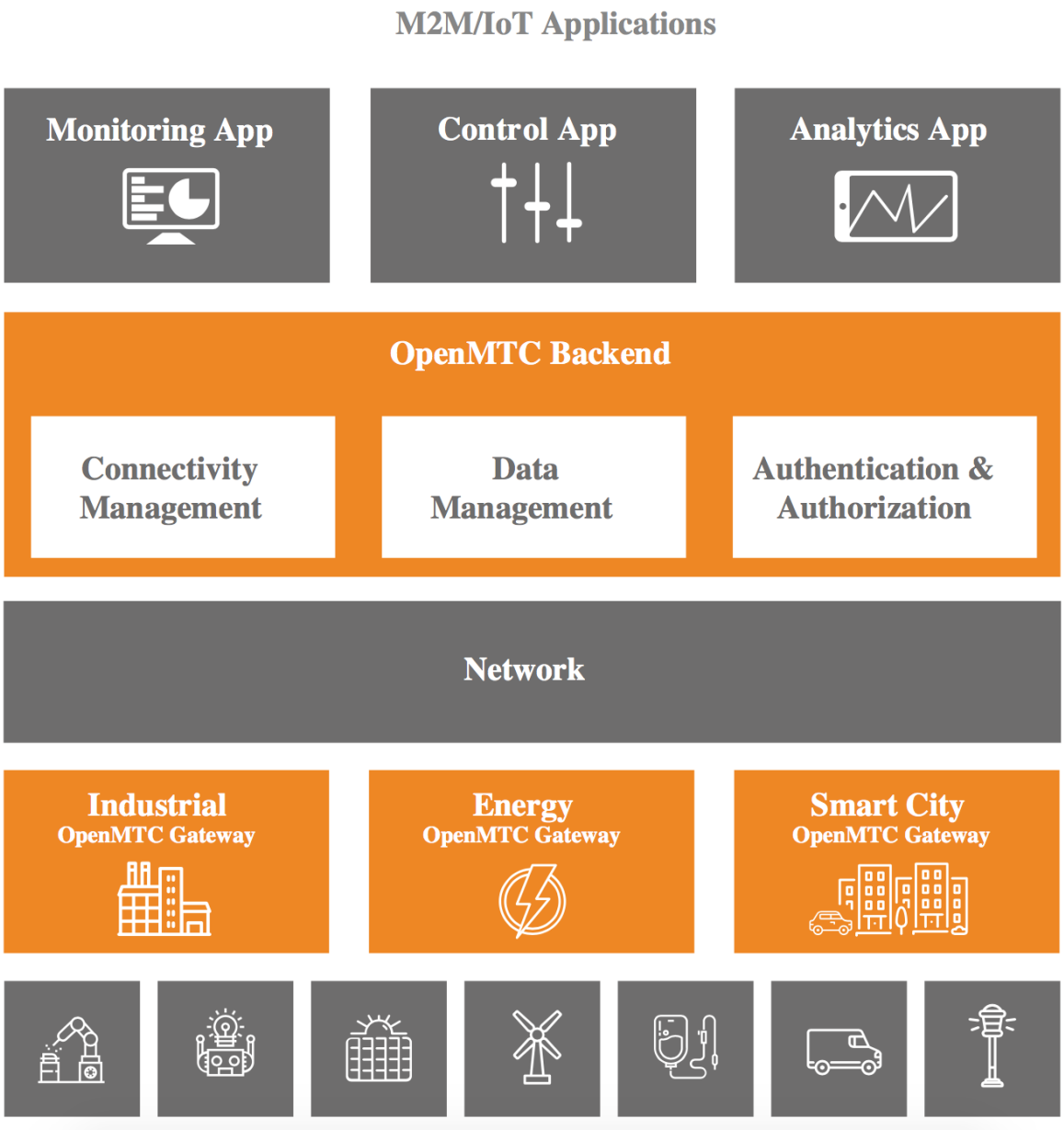

Figura 2.7: Plataforma OpenMTC[49]

HTTP, CoAP e *Websockets* para que diferentes áreas de domínio e aplicações possam interagir com a plataforma.

2.11.10 Carriots/Altair Smartworks

A *Carriots* é uma solução comercial baseada em PaaS voltada para aplicações M2M e IoT.

Para o desenvolvimento de aplicações, a plataforma disponibiliza o *Carriots App Engine*. É possível utilizar a linguagem de programação *Groovy* o que facilita o desenvolvimento de aplicações, fornecendo um ambiente de desenvolvimento simples. É também possível gerir regras e controlar a lógica de negócio através de um SDK.

A *Carriots* disponibiliza um módulo de gestão de dispositivos para: verificação do estado, alteração de configurações e atualização do *firmware*. Também fornece serviços de recolha de

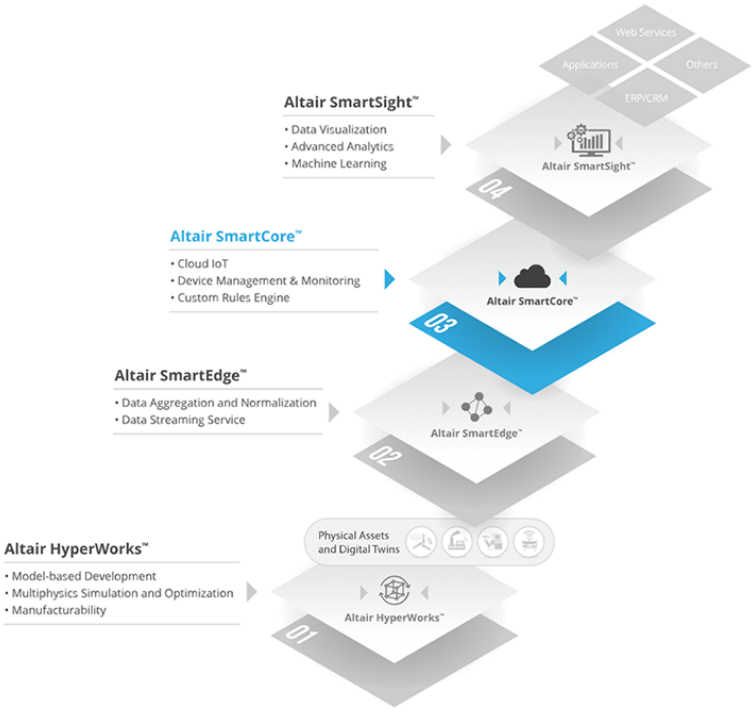

Figura 2.8: Plataforma Altair Smartworks[50]

dados e armazenamento.

Atualmente a plataforma *Carriots* designa-se por *Altair Smartworks*[50] como apresenta a Figura 2.8.

2.11.11 Xively

Xively é uma divisão da *LogMeIn Inc*, e é uma solução PaaS para a IoT. Define-se como uma "*cloud* pública especificamente construída para a IoT"[51]. Permite adicionar dispositivos, aplicações e serviços, conforme ilustrado na Figura 2.9.

Xively permite implementar e ligar de forma adequada e fácil um dispositivo na plataforma.

Após a sua integração, é possível gerir e interagir com os vários dispositivos através de SDKs e APIs disponibilizadas pela plataforma.

Esta plataforma disponibiliza um serviço para visualização de dados flexível e de alto desempenho, com a utilização de bases de dados temporais que permitem armazenar e recuperar dados. Também existe um sistema de *triggers* que permitem despoletar e executar em qualquer dispositivo ou aplicação ações avançadas.

Os dados gerados através dos diversos dispositivos podem ser utilizados para a criação de novas informações. Além disso é possível partilhar os dados com serviços externos como o Twitter, Facebook ou outros serviços.

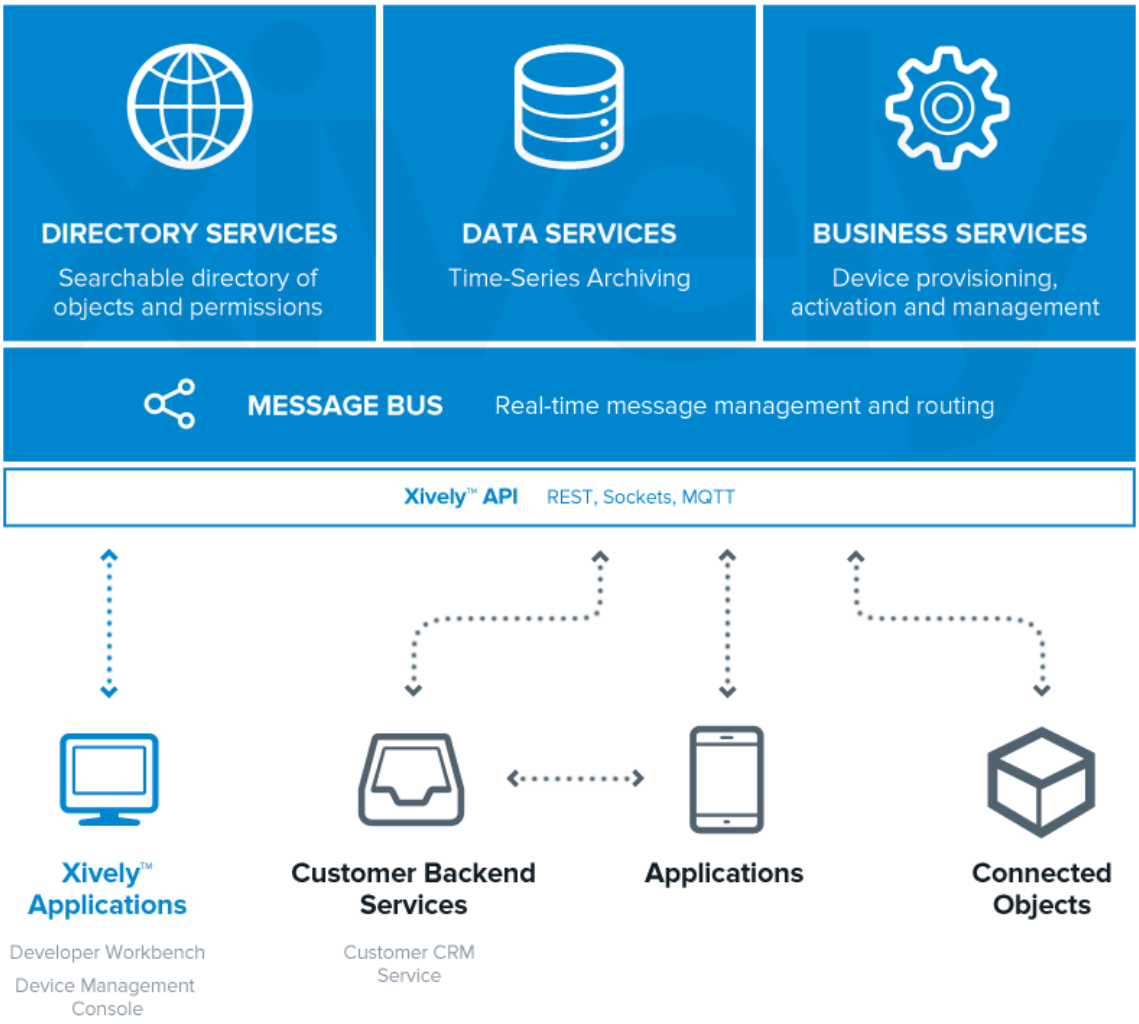

Figura 2.9: Xively Cloud Services[51]

2.11.12 Secção de Comparação - visão geral e comparação

Ao longo desta secção foram várias as abordagens analisadas que promovem a interoperabilidade entre plataformas. No entanto, a maioria delas não segue as mesmas abordagens.

Comparar, por exemplo, *Hypercat* e *Meshblu* não é apropriado, porque eles não têm muitas semelhanças. O *Hypercat* é uma plataforma munida de APIs, que permite a descoberta de dispositivos e manipulação dos dados daí provenientes, enquanto a *Meshblu* é um *broker* de mensagens que pretende tornar os dispositivos de baixo custo (sensores, principalmente) interoperáveis de maneira a que o protocolo de comunicação que eles usam torna-se irrelevante.

A *Ponte by Eclipse* pretende ser uma solução para receber e publicar dados usando múltiplos protocolos com suporte para uma variedade de formatos de dados.

Na tabela 2.1 será apresenta a comparação de algumas plataformas comerciais com uma avaliação de 1(baixo) a 4(alto) consoante o cumprimento de algumas características relevantes [52].

Das várias dimensões analisadas por Oleg Simakoff[52], *ThingWorx IoT Platform* obteve

	ThingWorx IoT Platform	Kaa IoT Platform	The Intel IoT Platform	thethings.io	IBM Watson Internet of Things Platform
Connectivity	4	4	4	4	2
MQTT Support	4	4	4	4	4
Telecom Integration	4	4	4	4	2
Machine Learning	4	4	4	4	3
Object Store	2	2	2	4	3
SDK	4	4	4	4	2
Audit	4	4	4	2	4
Access Management	4	4	1	1	4
Event Processing	4	4	4	4	4
Rules	4	4	4	4	3
Total	38	38	35	35	31

Tabela 2.1: Comparação de plataformas IoT[52]

38 pontos dos 40 possíveis à semelhança da plataforma open-source *Kaa IoT*. As plataformas *ThingWorx IoT Platform*, *Kaa IoT Platform* e *The Intel IoT Platform* obtiveram menor pontuação na capacidade de armazenamento, em contrapartida, a plataforma *thethings.iO* obteve pontuação máxima nesse parâmetro. Por outro lado, a plataforma *IBM Watson Internet of Things* apresenta algumas limitações no parâmetro *SDK* e *Connectivity*.

É óbvio que existe um longo caminho a percorrer até que a interoperabilidade seja total em arquiteturas M2M. Porém têm havido esforços e uma série de iniciativas que tentam resolver diferentes problemas nestes cenários.

CAPÍTULO 3

Descrição da arquitetura

O aparecimento de plataformas IoT permitiu que empresas e utilizadores interligassem os seus dispositivos com facilidade e a baixo custo. Para conseguir isso, existem algumas preocupações que precisam de ser abordadas, tais como: controle de acesso, interoperabilidade, criação de serviços e análise de dados.

A fim de obter um design mais genérico que potencialmente pudesse melhorar a interoperabilidade entre sistemas IoT, será apresentada uma proposta de arquitetura. Ao longo deste capítulo a arquitetura será descrita tendo por base os objetivos da sua criação: utilização de padrões recentes e interoperabilidade de multiplataformas IoT/M2M. No capítulo seguinte será apresentada uma implementação desta arquitetura.

A arquitetura foi desenhada focando-se em quatro pontos essenciais: 1) Controlo de Acessos; 2) Administração de cenários, dispositivos e sensores; 3) Visualização de dados; 4) Integração com outras plataformas.

Este capítulo pretende explicar como estes pontos foram abordados e dar uma visão detalhada da arquitetura. Durante este capítulo será apresentada a arquitectura na sua totalidade bem como cada um das partes que a compõem. Os detalhes da implementação serão abordados no próximo capítulo.

3.1 Descrição geral

A IoT deve estar "aberta" e todos os dispositivos e sensores devem poder comunicar entre si, mesmo que indiretamente. Um dispositivo deve ser capaz, por exemplo, de alertar outro dispositivo, que faz parte de um sistema IoT diferente, que algo mudou de estado e atuar sobre esse novo estado. Assim, muitos sistemas autónomos e inteligentes poderiam nascer para proporcionar uma qualidade de vida ainda melhor, uma vez que haveria uma coordenação mais inteligente e ampla dos dados e atuadores.

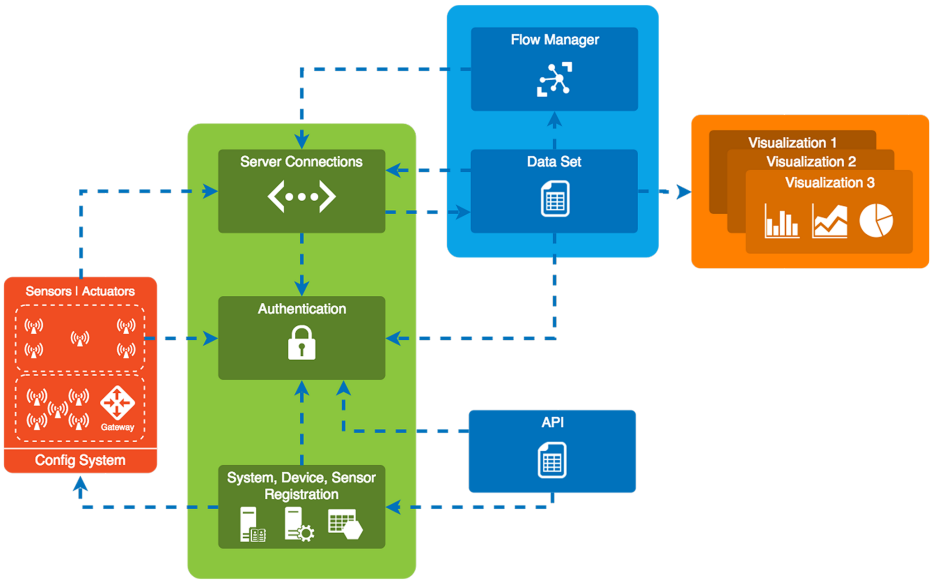

Figura 3.1: Arquitetura do Sistema

Com isso em mente, a Figura 3.1 ilustra as camadas da arquitetura e como estas podem interagir umas com as outras. A arquitetura proposta está dividida em 4 módulos distintos: *Sensor Network* e *Configuration System*, *Server Connections*, *Authentication* e *System, Device, Sensor Registration* e por fim o *Flow Manager* e respetiva *Data Visualization*.

A *Sensor Network* permitirá a comunicação entre diversos nós da rede de sensores e a sua configuração ficará a cargo do *Configuration System*. O *Server Connections* fará a ligação entre as redes de sensores e as várias aplicações *Visualization* e através do *Flow Manager* será possível filtrar e transformar os dados provenientes da *Sensor Network*. O módulo de *Authentication e System, Device, Sensor Registration* fará a gestão de utilizadores, definição de permissões e autorizações para cada utilizador no sistema. Também neste módulo será possível gerir sistemas/cenários, dispositivos e sensores. Por fim, será nos módulos *Flow Manager* e *Data Visualization* que será possível manipular e visualizar os dados tanto em tempo real como previamente armazenados através de acessos a uma base de dados.

Os próximos subcapítulos entrarão em maior detalhe cobrindo os módulos e componentes da arquitetura.

3.2 Sensor Network e Configuration System

A arquitetura é composta por uma WSN designada por *Sensor Network*. A *Sensor Network* poderá ser composta por um ou mais sensores independentes ou interligados através um dispositivo central (*gateway*) capaz de comunicar com os vários nós presentes nessa rede. Este nó "especial" recolhe os dados dos restantes sensores e encaminha-os para o *Server Connection*. As comunicações entre nós são realizadas através de RF ou métodos alternativos. Através do *Config System* é possível configurar os nós individualmente ou através do *gateway*, indicando as credenciais de acesso ao sistema e o identificador do nó/*gateway* previamente inserido no sistema.

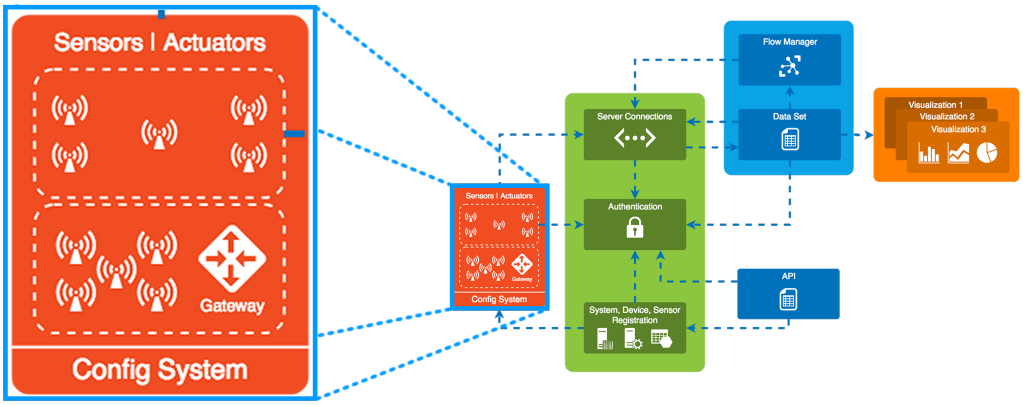

Figura 3.2: *Sensor Network* e *Configuration System*

Os dados ao serem enviados para o *Server Connections* ficam imediatamente disponíveis para monitorização em tempo real através de MQTT. Este protocolo foi escolhido, devido à sua eficiência e ao formato de comunicação das mensagens. Também é possível através do *Server Connections* transmitir os dados diretamente para uma aplicação *Web*, fazendo uso de *WebSockets*.

3.3 Server Connections

Na segunda camada, temos o *Server Connections* que serve como *broker* de mensagens. Este módulo permite a troca de mensagens instantâneas de máquina para máquina. A troca de mensagens realiza-se através do protocolo MQTT, porém também é possível a interação através de *WebSockets* que são protocolos amplamente utilizados na IoT.

O MQTT é um protocolo leve de mensagens *publish/subscribe* executado sobre o protocolo TCP/IP. Suas características tornam-no a escolha ideal para implementações de M2M em contexto de IoT.

O MQTT usa um padrão de mensagem *publish/subscribe* que permite uma distribuição de mensagens de um para muitos e um cliente pode subscrever vários canais. Estes canais podem ser subscritos a qualquer momento, bem como o cancelamento dos mesmos. Uma das maiores vantagens é o seu baixo *overhead*.

A autorização dos sensores e/ou *gateways* também é atendida pelo *broker*, que após consultar o módulo *Authentication* partilha as mensagens ou bloqueia os dados recebidos.

Os dados de autenticação são necessários a cada envio de mensagens. Se o dispositivo que executa a solicitação não puder ser identificado e autorizado, nenhum pedido será processado. Isso significa que, no momento da autenticação, os dispositivos serão responsáveis por armazenar o seu *token* para futuros envios de dados.

Além disso, o *Server Connections* permite a subscrição de mensagens enviadas a partir dos vários sensores, permitindo a sua monitorização. O *broker* é o componente que recebe as atualizações dos sensores e dos *gateways* e é responsável por autenticar quem enviou a mensagem e filtrá-los, de acordo com seu tipo.

Os clientes do *Server Connections* subdividem-se em *publishers*, *subscribers* e *processor*.

Um *publisher* lê os dados resultantes de uma *Sensor Network* diretamente a partir de um nó ou *gateway* e após um tratamento prévio disponibiliza essa informação num canal dedicado através do *Server Connections*.

Um *subscriber* fica à escuta de uma atualização dos dados disponibilizados pelo *publisher*. A cada receção de novos dados estes poderão ser manipulados, transformados e até armazenados.

Um *processor* é um elemento especial que consegue ser *subscriber* e *publisher* ao mesmo tempo. Desta forma é possível filtrar, manipular e realimentar o *Server Connections* com novas características e informações extraídas a partir dos dados originais provenientes das redes de sensores ou adicionando informações externas.

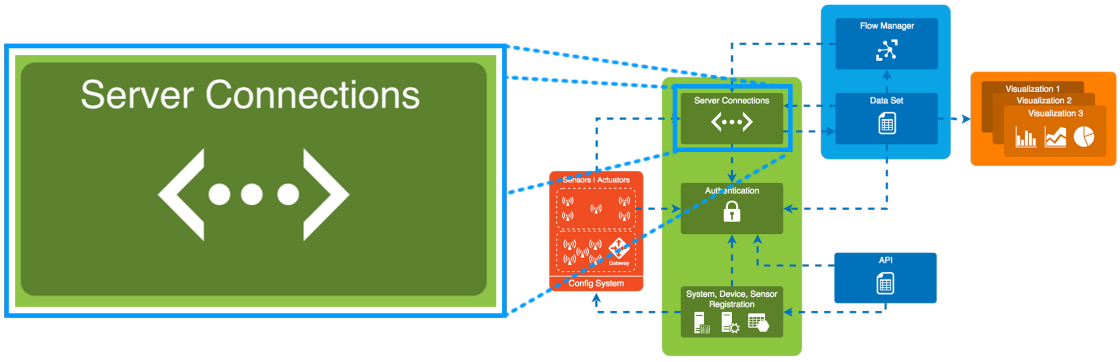

Figura 3.3: *Server Connections*

3.4 Authentication e System, Device, Sensor Registration

Este módulo da arquitetura permite registar, autenticar e posteriormente autorizar um utilizador registado no sistema através de um serviço *OAuth*. Através deste módulo é possível definir as permissões no sistema e de acesso aos dados de cada utilizador no sistema, ou seja, cada utilizador só verá os dados provenientes de uma *Sensor Network* própria ou partilhada com o mesmo.

Por outro lado, o utilizador poderá realizar a gestão dos vários nós das várias redes de sensores, bem como cenários da utilização dos sensores. Neste módulo é possível efetuar operações Create, Read, Update e Delete (CRUD) sobre os vários cenários/sistemas em utilização. Cada cenário/sistema poderá ter associado um ou mais dispositivos que por sua vez poderão ter associado um ou mais sensores. Após a introdução dos vários sensores na plataforma será possível a visualização de dados em tempo real e/ou dados previamente armazenados.

A informação é também acessível a partir de outras aplicações através da API. Esta permite a consulta dos vários sistemas, dispositivos e sensores.

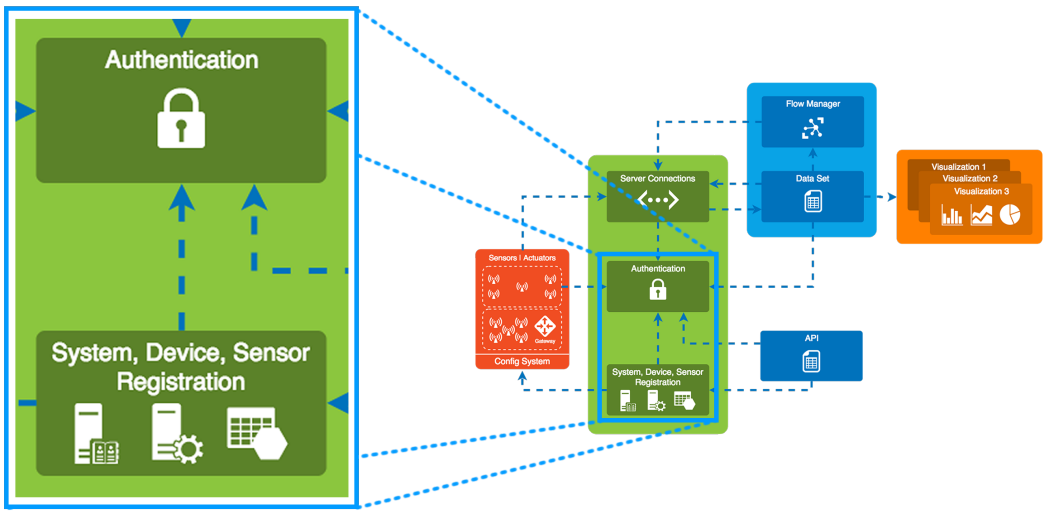

Figura 3.4: Authentication e System, Device, Sensor Registration

3.5 Flow Manager e Data Visualization

Por fim temos o módulo *Flow Manager* que permite a manipulação dos dados provenientes do *Server Connections*. Essas manipulações podem ser simples filtragens dos dados até operações de *machine learning*. Neste módulo é possível indicar que dados serão armazenados para futuras análises.

O módulo *Data Visualization* permite a visualização dos dados provenientes de uma *Sensor Network* ou dados já manipulados através de um fluxo de dados criado através do *Flow Manager*.

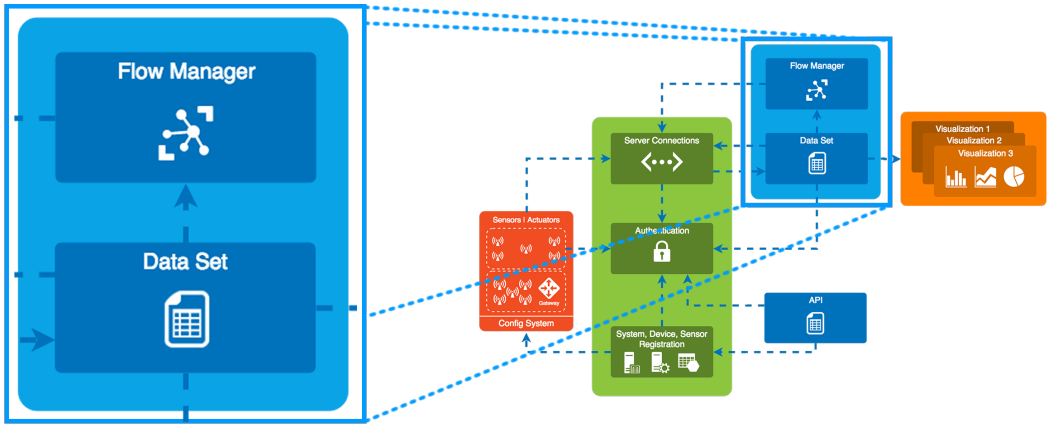

Figura 3.5: Flow Manager e Data Visualization

3.6 Conclusão

Foi objetivo deste capítulo apresentar todas as partes integrantes da arquitetura desenvolvida. Espera-se que esta aproximação permita alcançar a tão desejada IoT na sua plenitude. A IoT, devido a problemas de interoperabilidade, problemas de escalabilidade e falta de protocolos adequados para lidar com os padrões de comunicação e a segurança necessária, tem visto a sua adoção percorrer um processo lento e penoso.

A arquitetura enfatiza a importância da adoção de padrões IoT para facilitar a interoperabilidade de diferentes sistemas e incorporar diferentes protocolos para satisfazer e salvaguardar os requisitos de diferentes aplicações. Também é importante notar o suporte à monitorização na web em tempo (quase) real, pois demonstra a necessidade de os padrões IoT/M2M incorporarem ligações de protocolo adequadas ou, pelo menos, sejam flexíveis para permitir o desenvolvimento desse tipo.

A próxima secção detalhará toda a implementação da arquitetura, dando relevância ao *hardware* utilizado, módulos de *software* e paradigmas de comunicação entre os módulos que formam a solução final.

CAPÍTULO 4

Implementação e Testes

Um dos objetivos desta dissertação é a construção de um protótipo que demonstre a arquitetura desenvolvida. Assim, o protótipo pretende dar uma visão do que poderia ser uma aplicação suportada por esta arquitetura. Tanto os modelos de dados como o protótipo em si são simplistas e por isso meramente exemplificativos.

O cenário escolhido foi a agricultura de precisão, porém outros cenários seriam possíveis com as alterações estritamente necessárias. Este protótipo permite uma avaliação objetiva e periódica dos diversos fatores que afetam a produção agrícola.

A monitorização constante de vários parâmetros que afetam uma produção e a correção dos mesmos são dois fatores chaves no sucesso deste tipo de atividade. Com este projeto pretende-se implementar um protótipo funcional, com aquisição de dados utilizando uma rede sem fios de sensores e posterior processamento e apresentação de informação relevante. Em todo o caso, a aquisição dos dados pode ser monitorizada e visualizada em tempo real.

Na implementação foram utilizadas diversas plataformas de prototipagem tais como *Raspberry Pi*, *Arduino*, *ESP8266*, *ESP32* e equipamentos/soluções profissionais, atualmente em comercialização, e desenhadas especificamente para o efeito como é o caso da *Libelium*.

Numa primeira fase será apresentada a solução testada bem como os vários módulos da arquitetura, de seguida serão apresentadas as tecnologias e plataformas utilizadas no desenvolvimento, terminando com uma série de resultados provenientes de alguns *Benchmarks*.

4.1 Solução testada

Como um meio para mostrar o potencial da arquitetura apresentada no Capítulo 3 foi desenvolvido um sistema IoT utilizando padrões e protocolos adequados a fim de monitorizar uma solução de agricultura de precisão.

A partir de um conjunto de sensores devidamente inseridos na *System, Device, Sensor Registration*, configurados utilizando o *Configuration System*, os dados presentes na *Sensor Network* são enviados através de soluções sem fio até ao *Server Connections*. A partir desse momento poderão existir *subscribers* devidamente autorizados através do módulo *Authentication* a consumir esses dados. Assim, em tempo real, é possível consultar, manipular e criar novas informações, bem como criar simplesmente alertas de possíveis alterações.

Para a visualização das representações dos vários sensores é possível utilizar o *Data Set*, consultando em tempo real as alterações dos mesmos através das diferentes *Visualization*, sem que sejam necessárias configurações adicionais, pois todos os sensores são representados dinamicamente tendo por base as suas configurações. Por outro lado, é possível criar novas informações, bem como guardar, filtrar e manipular os dados recebidos através do *Flow Manager*.

Tanto a API do *System, Device, Sensor Registration* como o próprio *Server Connections* podem ser utilizados por terceiros bastando para tal que apresentem autorizações válidas.

Na Figura 4.1 é possível verificar as várias tecnologias de comunicação utilizadas ao longo da conceção do protótipo.

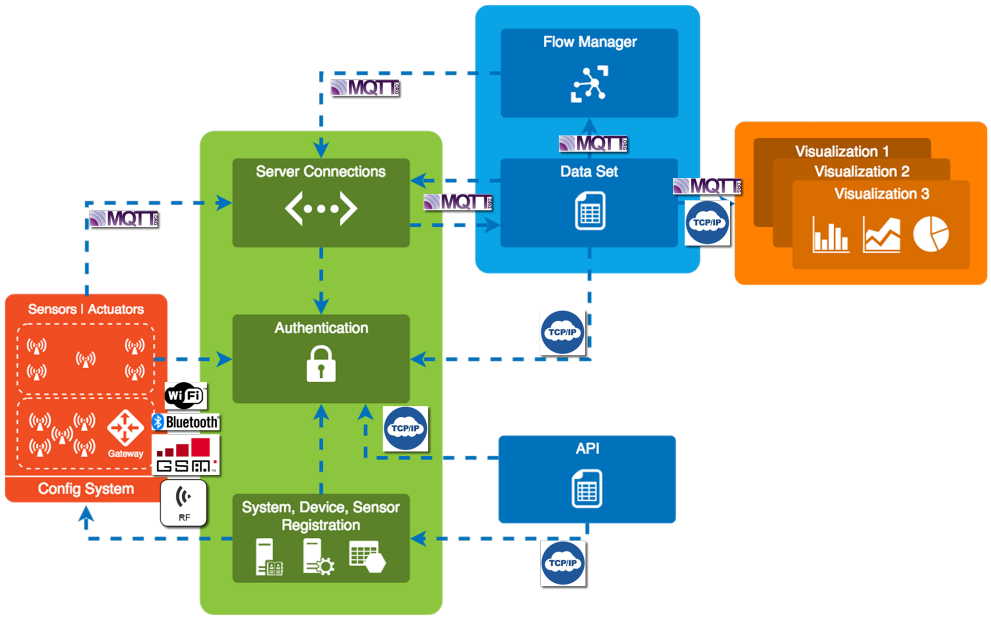

Figura 4.1: Tecnologias de comunicação

Do ponto de vista tecnológico, há um conjunto de aspetos que devem ser considerados para este tipo de cenário: Interoperabilidade, Escalabilidade, QoS e segurança.

A interoperabilidade no IoT desempenha um papel importante, pelo que é necessário garantir que a solução desenvolvida seja compatível com os padrões IoT mais recentes. A

própria *Sensor Network* também deve lidar com a escalabilidade. Esta deve ser capaz de lidar com uma maior quantidade de dispositivos, manipulando adequadamente a carga de tráfego por meio da adoção de protocolos eficientes e capacidades de balanceamento de carga. Devemos ter em consideração a QoS e segurança, pois, necessitamos de ter a certeza de que a rede é confiável e que entregamos todos os dados o mais rapidamente possível (monitorização em tempo real), protegendo a privacidade do utilizador.

Para que seja possível a execução deste protótipo, os serviços presentes no Anexo A terão de ser executados.

De seguida, a arquitetura do sistema é apresentada com todas essas questões tidas em consideração. Todas as suas partes são cuidadosamente explicadas nas secções seguintes.

4.2 Sensor Network e Configuration System

O cenário de aplicação deste projeto é a agricultura e por esse motivo os sensores utilizados estão relacionados direta ou indiretamente com essa atividade. Esses sensores, por sua vez, estão (ou podem estar) ligados a um dispositivo central denominado *gateway*.

Sensores	**WaspMote Agricultura**	**WaspMote Gases**	**WaspMote SmartCities**	**Arduino / ESP8266 / ESP32**
Temperatura Solo	X			X
Temperatura Ar	X		X	X
Humidade Solo	X			X
Humidade Ar	X			X
Velocidade do Vento	X			
Direção do Vento	X			
Precipitação	X			
Gases NO		X		
Gases CO		X		
Gases CO2		X		
Radiação Solar		X		
Deteção de movimento			X	X

Tabela 4.1: Sensores atualmente utilizados

Na Tabela 4.1 e Figura 4.2 podem ser vistos alguns dos sensores utilizados ao longo do desenvolvimento. Inicialmente, na *Sensor Network* foram utilizados alguns kits WaspMote da Libelium[53]. Estes kits temáticos contêm sensores associados a agricultura, *Smart Cities*, análise de gases, entre outros que gradualmente foram introduzidos na aplicação da arquitetura descrita anteriormente.

Durante a implementação foram adicionadas placas de desenvolvimento *ESP8266* e *ESP32*[54] munidos de diversos sensores. Utilizando esta placa de desenvolvimento foi necessário adaptar o *Server Connection* para que permitisse comunicações *Wi-Fi* com a *Sensor Network*. Desta forma, foi possível que o sistema de comunicações implementado permitisse um conjunto diverso de protocolos de comunicação, tais como RF, permitindo assim comunicações até cerca de 1km de distância, Wi-Fi e Groupe Special Mobile (GSM).

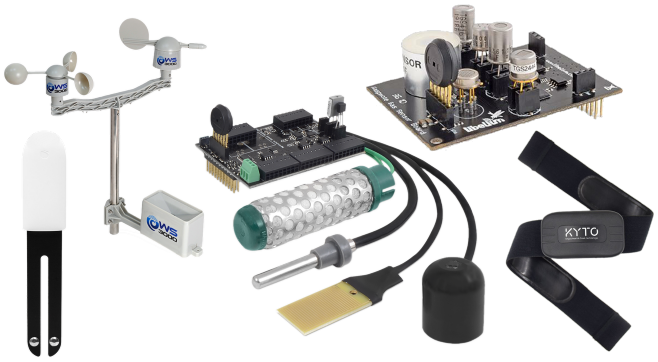

Figura 4.2: Alguns dos sensores utilizados

Todos os sensores são configurados, através do *Configuration System* do sensor ou do *gateway* do conjunto de sensores. Através do *Configuration System* são inseridos todos os parâmetros referentes aos sensores. Este módulo tem por base um *Captive Portal*. Também é possível realizar atualizações de software remotamente através de um serviço *OTA*.

Depois de configurados, os sensores têm como principal função enviar dados resultantes das suas leituras para o *Server Connection*. Esses dados são enviados através de diversos protocolos dependentes da sua interface de comunicação. Por outro lado, existem também atuadores que respondem a comandos enviados a partir da plataforma.

Assim, temos sensores que comunicam diretamente com o *Server Connections*, porém outros possuem apenas uma interface de comunicação de RF que envia os seus dados para um *gateway* que, por sua vez, os converte em pacotes UDP que são enviados para um servidor UDP. O servidor UDP recebe também os dados provenientes de outros sensores com capacidade de envio de pacotes UDP, como por exemplo os sensores *Wi-Fi* e os sensores GSM. O servidor UDP comunica depois com o *Server Connection* através de MQTT, demonstrando mais uma vez a versatilidade do *Server Connection*.

Nos módulos desenvolvidos utilizando o *Arduino/ESP8266/ESP32* foi necessário, inicialmente, criar uma rede de sensores e definir um protocolo de tramas, que seriam trocadas entre os vários nós da rede. Recorreu-se a marcadores de início e fim de trama, para garantir a boa receção da trama. Compreendido entre estes marcadores seguem o endereço de origem e de destino, tamanho da trama e um *checksum* para garantir que a mensagem será recebida corretamente. Nestes módulos foi utilizado um *Arduino* como placa de desenvolvimento, um módulo RF e um ou mais sensores tais como: sensores de temperatura e humidade do solo e do ar, Light Dependent Resistor (LDR), GPS, acelerómetro, giroscópio, bússola e sensores presentes numa estação meteorológica.

4.2.1 Captive Portal

Um *Captive Portal* é uma página web que é exibida para novos utilizadores de uma determinada plataforma antes mesmo de terem acesso aos diversos recursos. *Captive Portals* são usualmente usados para apresentar uma página de *login*, que pode exigir autenticação, pagamento, aceitação de End-User License Agreement (EULA) ou outras credenciais antes mesmo da

utilização da plataforma. Estes recursos são encontrados frequentemente em serviços de banda larga móveis e *hotspots* públicos. Um *Captive Portal* pode ser também utilizado em pontos de acesso a redes com fio corporativas ou residenciais, como apartamentos e quartos de hotel.

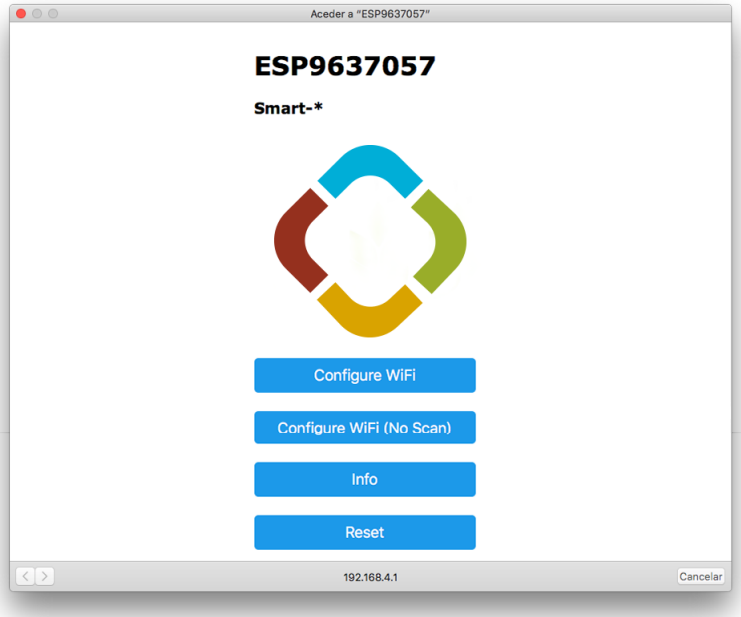

Figura 4.3: Config System - Configuração do equipamento

Os *Captive Portals* são usados principalmente em redes sem fio abertas, onde os utilizadores recebem uma mensagem de boas-vindas informando-os sobre as condições de acesso.

Frequentemente, *Captive Portals* são usados para fins de marketing e comunicação comercial. O acesso à *Internet* por *Wi-Fi* aberto é proibido até que o utilizador preencha um formulário de registo.

No que diz respeito à implementação da arquitetura proposta, a funcionalidade *Captive Portal* permite configurar um sensor ou rede de sensores através do *gateway* respetivo. Assim, através da introdução das credenciais e outras informações importantes é possível associar um sensor ao respetivo utilizador como pode ser visto pelas Figuras 4.3 e 4.4.

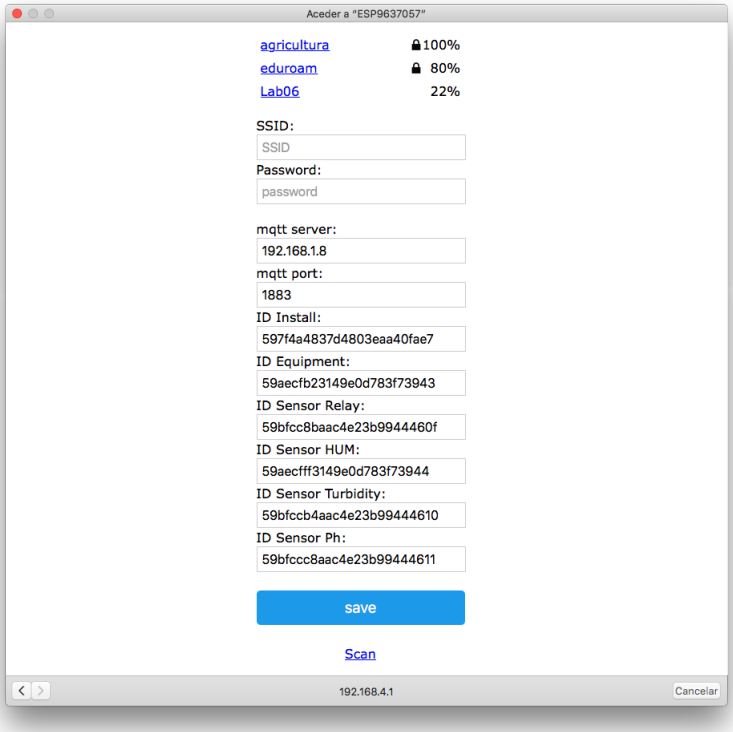

Figura 4.4: Config System - Configuração dos vários sensores

4.2.2 OTA

A programação Over-the-Air (OTA) refere-se a vários métodos de distribuição de novos softwares e configurações. Uma característica importante do OTA é que a partir de um local central é possível enviar uma atualização para todos os equipamentos.

Em dispositivos móveis, como smartphones, uma atualização via OTA pode se referir simplesmente a uma atualização de software distribuída por *Wi-Fi* ou banda larga móvel. Desta forma não é necessário conectar o dispositivo a um computador via Universal Serial Bus (USB) para realizar a atualização. As atualizações de firmware encontram-se simplesmente disponíveis para download através de um serviço OTA.

Mais recentemente, com os novos conceitos de WSN e IoT, onde as redes consistem em centenas ou milhares de nós, o OTA é de extremamente importância para a transmissão de atualizações para os diversos nós que se encontram em locais remotos ou em difícil acesso, utilizando para o efeito comunicações em bandas de frequência não licenciadas (868 MHz , 900 MHz, 2400 MHz) e com baixo consumo e baixa taxa de transmissão de dados usando protocolos como 802.15.4 e *ZigBee*.

Como exemplo, a *Libelium* implementou um sistema OTA inteligente e fácil de usar para dispositivos *ZigBee* WSN. Este sistema permite atualizações de firmware sem a necessidade de acesso físico, economizando tempo e dinheiro se os nós necessitarem de ser reprogramados. Esta funcionalidade foi também implementada na *Sensor Network*.

4.3 Server Connections

O módulo *Server Connections* é responsável por receber os dados provenientes das redes de sensores e disponibilizá-los em canais de comunicação que poderão ser acedidos pelos restantes módulos da arquitetura. Foi implementado em *Node.js* tendo por base um broker MQTT, *Mosca*. O *Mosca*[55] permite uma personalização completa da implementação da arquitetura *publish-subscribe* do MQTT. MQTT[56] (originalmente chamado *MQ Telemetry Transport*) é um protocolo de mensagens M2M e IoT projetado como um protocolo extremamente leve o que é uma mais-valia na implementação desta arquitetura. Este protocolo tem um *overhead* de dados extremamente baixo (cerca de dois *bytes*) e tem suporte para diferentes níveis de QoS. A capacidade de personalização do *Mosca* é uma mais-valia, visto que dessa forma é possível alterar os métodos *authorizePublish* e *authorizeSubscribe*. Dado que se pretende que esta solução suporte, em simultâneo, diversas redes de sensores, de utilizadores distintos, a alteração dos métodos referidos permite, assim, garantir que só as pessoas autorizadas poderão visualizar os dados provenientes da *Sensor Network* com a qual estão relacionados.

Foi também desenvolvido uma interface através de *WebSockets* para permitir a monitorização *web* em tempo real. Através desta extensão é possível estender a capacidade de suporte a outras aplicações IoT que não interliguem diretamente através de MQTT.

4.4 Authentication e System, Device, Sensor Registration

Os módulos de *Authentication* e *System, Device, Sensor Registration* estão responsáveis pela autenticação dos utilizadores no sistema e pelo registo dos cenários, dispositivos e sensores associados à plataforma. Através deste módulo centralizado é possível registar utilizadores, definir níveis de permissões sobre ações no sistema e gerir níveis de autorização sobre os dados no *Server Connection*.

O módulo de autenticação foi desenvolvido em *Node.js* utilizando o *middleware Passportjs*. O *Passportjs*[57] está responsável por autenticar solicitações que têm origem no *Flow Manager*, *Data Set* e respetiva API. Este implementa *OAuth* para a autenticação através do par utilizador/*password*, bem como através de redes sociais ou outros sistemas baseados na troca de *tokens* para proteção dos utilizadores. Essa separação da autenticação da restante lógica de negócio mantém o código limpo e torna o *Passportjs* extremamente fácil de integrar numa aplicação.

Em aplicações *web*, a autenticação pode assumir uma variedade de formas. Tradicionalmente, os utilizadores efetuam o *login* fornecendo um nome de utilizador e uma *password*. Com o surgimento das redes sociais e utilizando um *single sign-on* através de um *provider OAuth*, como Facebook ou Twitter, tornou o *OAuth* num método de autenticação popular. Por exemplo, serviços que disponibilizam uma API geralmente exigem credenciais baseadas num *token* para acesso às suas informações e para tal utilizam *OAuth*, tal qual acontece na API presente no sistema.

O *Passportjs* é extremamente modular e dessa forma consegue adaptar-se a qualquer aplicação, criando por vezes soluções exclusivas de autenticação. Os mecanismos de autenticação

são conhecidos geralmente como estratégias que são desenhados como módulos individuais. Desta forma, as aplicações podem escolher que estratégias irão utilizar, sem criar dependências desnecessárias a outras estratégias não utilizadas na aplicação.

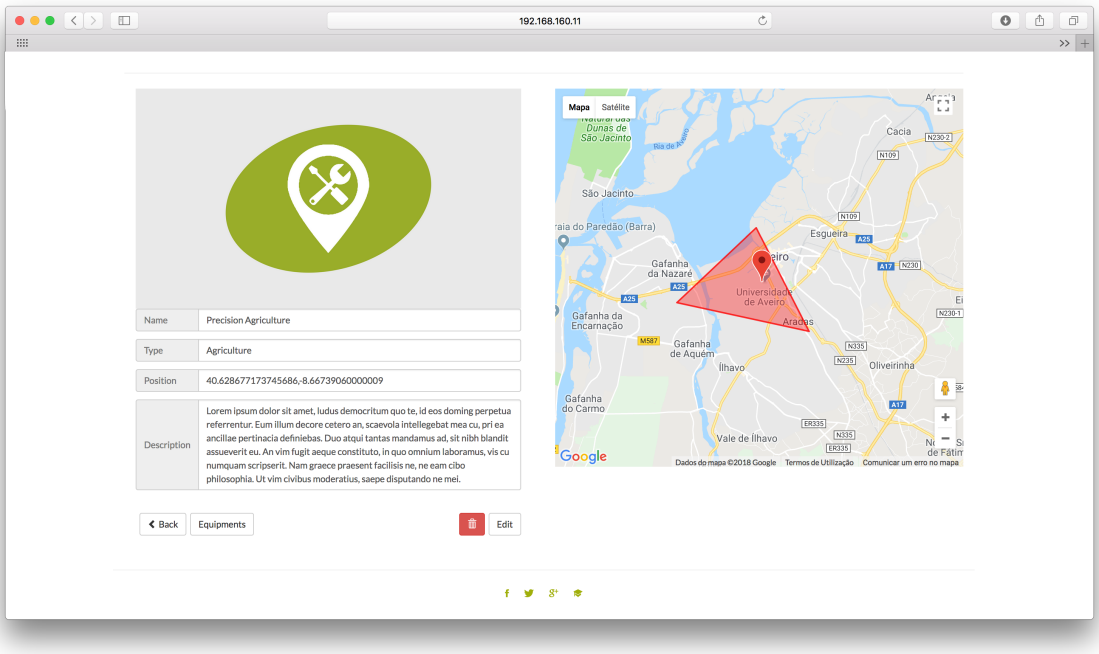

Figura 4.5: Exemplo da interface do protótipo

O módulo *System, Device, Sensor Registration* permite fazer a gestão dos cenários de utilização e dos vários dispositivos que contêm sensores. Assim é possível, por exemplo, definir modelos de visualização por defeito para um determinado sensor, definir *thresholds* para os dados provenientes da *Server Network* ou gerir os vários dispositivos presentes num determinado cenário.

Afim de armazenar os dados provenientes do módulo *System, Device, Sensor Registration* foi elaborado um modelo de dados presente no Anexo B.

Após efetuar a autenticação no sistema, o utilizador poderá gerir todos os cenários utilizando as tradicionais operações CRUD que permitem realizar a gestão dos sensores.

Para além da interface gráfica (Figura 4.5) existe uma API que permite a consulta dos vários componentes deste cenário que se encontra acompanhada pela sua documentação como poderá ser visto através da Figura 4.6.

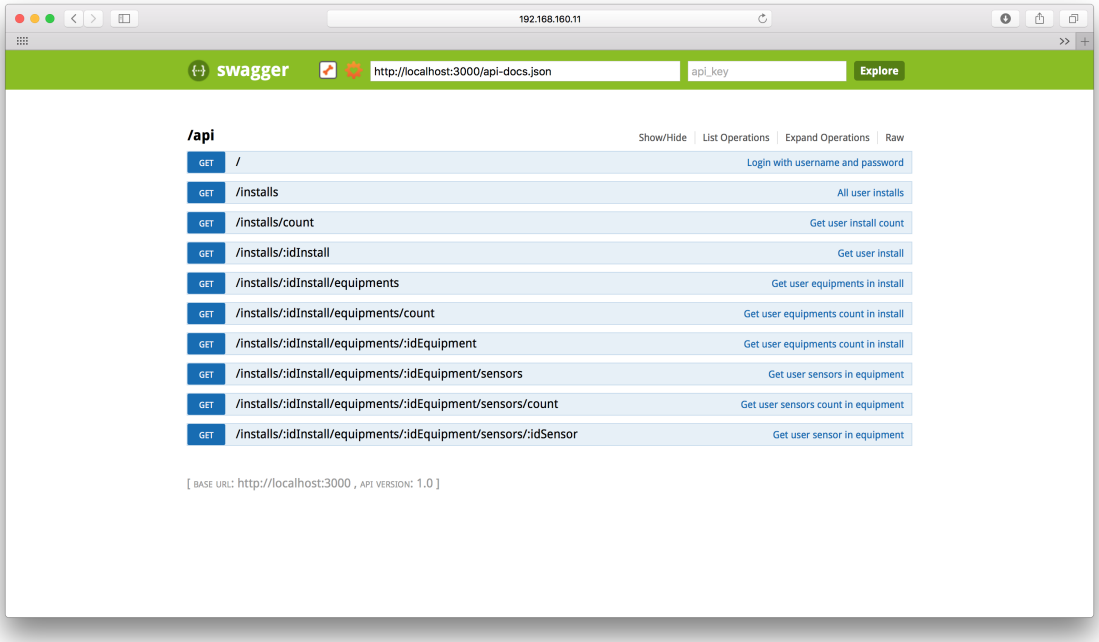

Figura 4.6: Definição da API disponibilizada

4.5 Flow Manager e Data Set

O *Flow Manager* é o orquestrador dos dados provenientes das *Sensor Network*. Este módulo foi baseado em *Node-RED*[58], plataforma desenvolvida em *Node.js* para a interligação de nós na IoT. Através deste é possível definir o fluxo dos dados arrastando para a área de trabalho os nós associados aos sensores bem como a forma como os dados dos mesmos serão apresentados na interface. Assim o *Flow Manager* permite ao utilizador definir como pretende receber, manipular e visualizar os dados.

Inicialmente o *Flow Manager* é criado dinamicamente a partir dos dados presentes no módulos de *Authentication* e *System, Device, Sensor Registration* através de uma solução de *templates* e a utilização do *mustache.js*[59]. Mais informações poderão ser consultadas no Anexo C.

Associado ao *Flow Manager* foram desenvolvidos alguns *publishers/subscribers* que foram apelidados de *processings*. Um *processing* é um elemento especial que consegue ser *subscriber* e *publisher* ao mesmo tempo. Desta forma é possível filtrar, manipular e realimentar o *Server Connections* com novas características e informações extraídas a partir dos dados originais. O utilizador define o fluxo dos dados provenientes dos sensores registados anteriormente, manipulando, filtrando e por fim personalizando a sua visualização.

Este orquestrador foi criado através de uma solução de *plugins* que consoante a temática poderá contar com nós específicos para a aquisição e manipulação dos dados como pode ser visto na Figura 4.7, demonstrando uma grande versatilidade.

Durante esta dissertação foram criados conjuntos de nós associados a Agricultura de Precisão, Smart Cities e Desporto, respetivamente, *Node-RED-contrib-precAgriNodes*, *Node-*

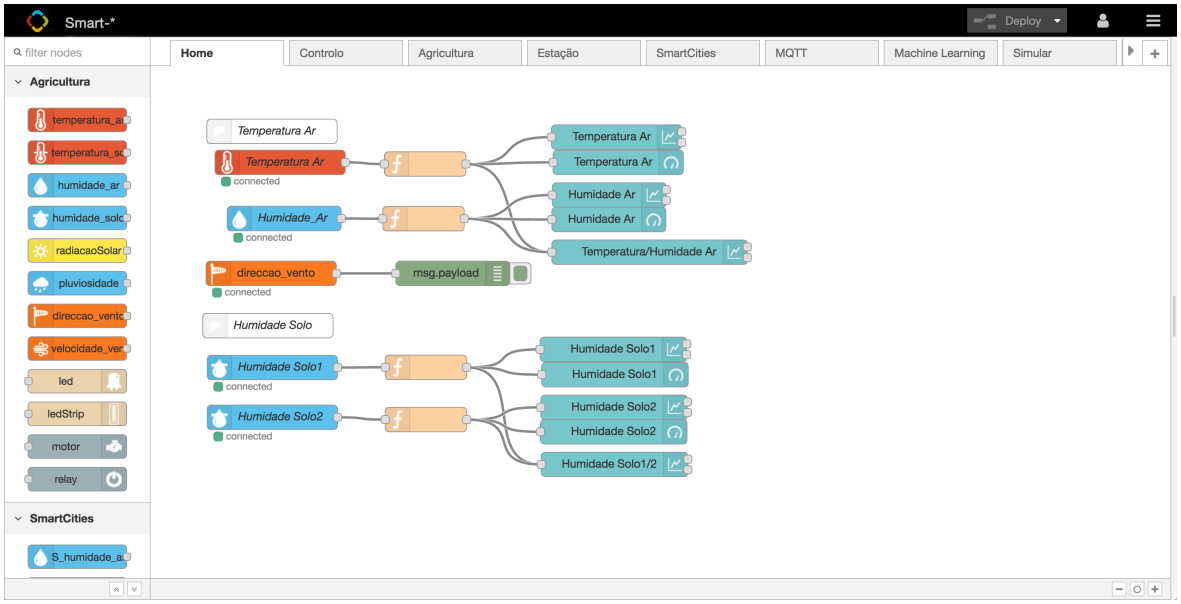

Figura 4.7: Flow Manager

Figura 4.8: Visualização de dados personalizada pelo utilizador

RED-contrib-smartCitiesNodes, *Node-RED-contrib-sportNodes*. Deste conjunto de nós podemos destacar os nós representantes de sensores de leitura de CO, CO_2, NO_2, direção do vento, humidade do solo e do ar, luminosidade, radiação solar, temperatura do solo e do ar, ultra-som, velocidade do vento, pluviosidade, poeira e pressão. Nestes *plugins* também constam representações de atuadores como leds, motores e bombas de água, por exemplo.

Um novo *plugin* pode ser criado e adicionado a partir do tutorial presente no Anexo D.

Outra área de extrema relevância é o *Visualization*. Esta visualização personalizável e

definida através do orquestrador.

A visualização dos dados provenientes dos sensores e sua monitorização é possível através do *Data Set*. Nesta aplicação também está contemplada a possível distribuição dos dados com aplicações terceiras através deste mesmo módulo.

A Figura 4.8 é a visualização dos dados anteriormente criada na Figura 4.7. Desta forma está constatada a capacidade de personalização, adaptação e liberdade na visualização de dados em tempo real disponibilizados pelo sistema.

Tanto os fluxos no *Flow Manager* como a visualização dos dados no *Data Set*, à partida são gerados automaticamente, podendo, em todo o caso, o utilizador personalizar toda a visualização. Para além destes, é possível exportar o cenário para uma implementação no *home-assistant* como pode ser visto através da Figura 4.11.

4.5.1 Monitorização em tempo real

Num sistema que utiliza um arquitetura cliente-servidor tradicional, os sensores fazem a recolha dos dados, enviam-na para um servidor que a guarda numa base de dados e depois uma aplicação acede a essa base de dados e trata os dados de forma adequada à sua representação.

O sistema, através do *Server Connection*, permite monitorizar um sensor ou conjunto de sensores e disponibilizar os dados num determinado canal de comunicações. Os dados que circulam por esse canal poderão ser representados diretamente numa interface personalizada pelo utilizador.

Assim sendo, a possibilidade de acompanhamento e análise em tempo real de dados e posterior análise é uma mais valia para gestores e analistas dos dados provenientes destas redes de sensores.

4.5.2 Alarmística

Os *processings*, para além de receberem dados dos sensores e de tratarem do seu envio para o *Server Connection* para posterior armazenamento em base de dados, têm ainda uma função de análise da qualidade dos dados recebidos. Essa função de análise está posteriormente associada com um serviço de alarmística disponível no *Flow Manager*.

Para exemplificar o procedimento de funcionamento de uma alarmística, vamos supor o caso de um *processing* dedicado à recolha de temperatura do ar. No *Flow Manager* são definidos conjuntos de valores:

- Valor normal: entre 10 e 35 graus centígrados;
- Valor anormal: entre os 0 e os 9 graus ou entre os 36 e os 45 graus centígrados;
- Possivelmente avariado: fora destas gamas de valores.

Assim, se uma leitura de valor for, por exemplo 37 graus centígrados, uma mensagem de alarme é enviada e poderá visualizar na interface essa mesma indicação. Esta mensagem é enviada por um *processing* específico, pelo que qualquer *Data Set* (além da interface de visualização criada nesta prova de conceito) poderá usar tal informação, desde que tenha permissões para tal.

4.6 Tecnologias e plataformas utilizadas

Para o desenvolvimento deste protótipo, foram usadas várias tecnologias, ferramentas e linguagens de programação, algumas já referidas anteriormente aquando da descrição dos módulos utilizados como pode ser visto através da Figura 4.9. Para além das tecnologias utilizadas e já descritas, existem outras que serão apresentadas de seguida.

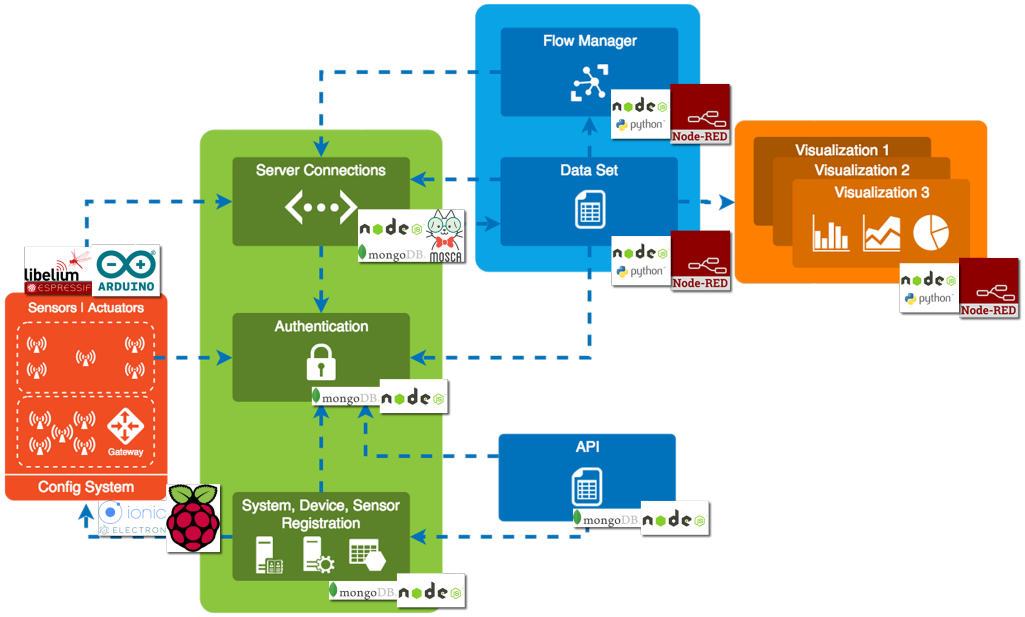

Figura 4.9: Tecnologias da implementação

4.6.1 Backend

Para a criação do *backend*, a linguagem de programação utilizada foi *Node.js*.

Node.js é uma plataforma para desenvolvimento de aplicações *server-side* baseadas em *JavaScript* e o *V8 JavaScript Engine*, ou seja, com Node.js podemos criar uma variedade de aplicações *web* utilizando apenas código em *JavaScript*.

Node.js permite a criação de soluções escaláveis de alta performance, leves, rápidas e poderosas. Além disso, utiliza uma arquitetura baseada em eventos. Usando um *loop* de eventos, o *Node.js* interpreta, em uma única *thread*, as requisições de forma assíncrona em vez de sequenciais, não permitindo bloqueios. Afim de facilitar o processo de desenvolvimento foi adotada uma *framework* de aplicações *web* para *Node.js*, *Express.js*. Esta foi projetada para construir aplicações *web* e APIs.

Associado a este módulo foi criado um sistema de autenticação implementado tendo por base o *Passport.js*.

Ao longo do desenvolvimento do protótipo, foram utilizados alguns módulos de forma a permitir agilizar o sistema e simplificar alguns passos. Seguidamente, serão apresentadas alguns das principais tecnologias utilizadas.

Passport.js

Passport.js é um *middleware* para *Node.js* que implementa o módulo de autenticação em uma aplicação de forma rápida e fácil. É possível implementar numa única aplicação *web* vários mecanismos de autenticação para além do habitual par utilizador/*password*.

Com o crescimento das redes sociais, a utilização de *providers OAuth* têm-se tornado bastante populares através da utilização do *Facebook* ou o *Twitter*.

Passport.js reconhece que cada aplicação tem requisitos únicos de autenticação. Mecanismos de autenticação, conhecidos como estratégias são disponibilizados como módulos individuais. É possível utilizar mais de 140 mecanismos de autenticação e todas essas estratégias são independentes umas das outras e adicionadas como *plugins* à aplicação *web*.

MongoDB

MongoDB é uma base de dados multi-plataforma Not only SQL (NoSQL), escrita em *C++* e disponibilizada como *open-source*.

Pode ser executado em *Windows*, *Linux* e *MacOS*, com integração nas linguagens de programação mais populares, como *C#*, *Java*, *PHP*, *Node.js* e *Python*.

MongoDB é um base de dados orientada a documentos que armazena os dados em documentos não estruturados. Isso significa que é possível armazenar registos sem que seja definida uma estrutura de dados previamente, como o número de campos ou tipos de campos para armazenar valores. Os documentos do *MongoDB* são semelhantes aos objetos JSON.

MongoDB é um base de dados multi-plataforma NoSQL, que pode ser executado em *Windows*, *Linux* e *MacOS* e permite a integração nas linguagens de programação mais populares, como *C#*, *Java*, *PHP*, *Node.js* e *Python*.

Habitualmente numa Relational Database Management System (RDBMS) os dados são armazenados em tabelas formatadas e consultadas através de Structured Query Language (SQL). No caso do *MongoDB*, este guarda os dados em documentos ao invés de tabelas, assim é possível alterar a estrutura dos documentos simplesmente adicionando novos campos ou excluindo os existentes sem comprometer a informação anteriormente adicionada. Esta capacidade do *MongoDB* é útil para representar relações hierárquicas, armazenar matrizes e outras estruturas mais complexas de uma forma mais simples. *MongoDB* oferece alto desempenho, alta disponibilidade, fácil escalabilidade através dos serviços de *replicaSet* e *sharding*.

GeoJSON

Durante o desenvolvimento da aplicação foi decidido que seriam utilizados dados geográficos.O formato escolhido foi o *GeoJSON*.

GeoJSON é um formato padrão aberto projetado para representar características geográficas baseadas em JSON.

TopoJSON é uma extensão do *GeoJSON* que codifica a topologia geoespacial. Em vez de representar discretamente geometrias, estas são unidas a partir de segmentos de linha compartilhados chamados de *arcs*. Permite que as geometrias relacionadas sejam armazenadas

eficientemente no mesmo arquivo, eliminando assim a redundância. Como resultado disso, o *TopoJSON* é substancialmente mais compacto que o *GeoJSON*, conseguindo uma redução de cerca de 80,4% [60].

Este formato pode descrever formas vetoriais como *points*, *line strings* e *polygons* que podem representar, por exemplo, terrenos agrícolas, rios e lagos. Cada uma destas formas pode conter atributos que o descrevem (ex.: nome, área, etc.). As formas juntamente com os seus atributos podem criar infinitas representações sobre dados geográficos. Tais representações fornecem a possibilidade de uma representação específica e precisa.

4.6.2 Frontend

Para a criação do *frontend*, isto é, a interface do utilizador, foi utilizado *HTML5*, *CSS* e a *framework Bootstrap*. *Boostrap* é uma *web framework* gratuita que consiste num conjunto de ferramentas para a criação de aplicações *web* permitindo uma interface responsiva.

Também é utilizado *JavaScript* em conjugação com a biblioteca *jQuery*. Esta biblioteca adiciona funcionalidades que permitem um código mais simplificado e compatibilidade entre todos os diferentes *browsers*.

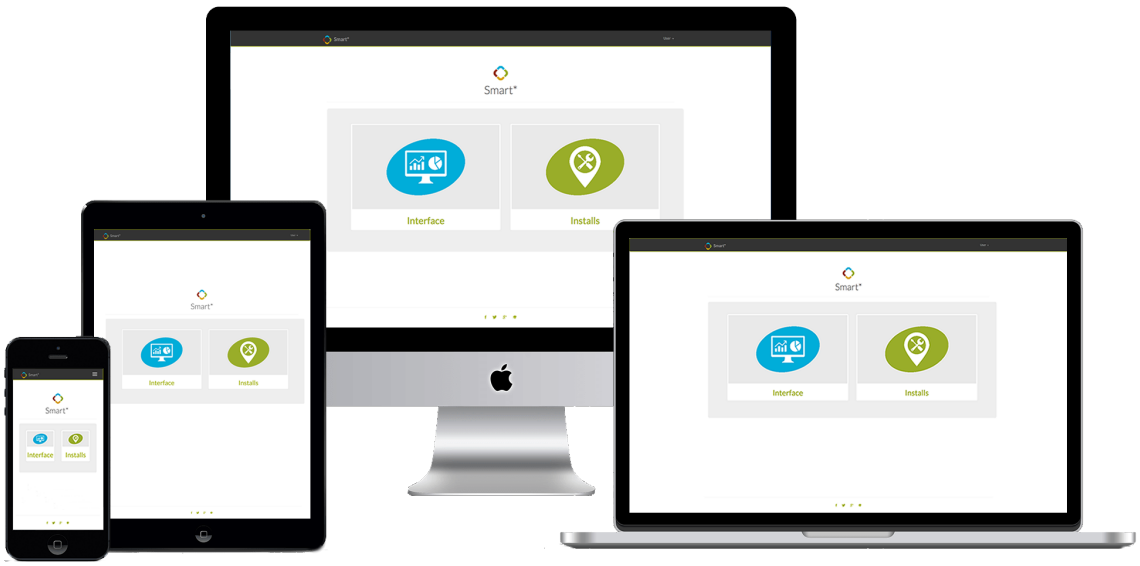

Figura 4.10: Utilização do protótipo nas diversas plataformas

No Anexo E está o manual de utilização do protótipo (Figura 1) para que seja possível perceber a organização e principais funcionalidades.

HTML5

O *HTML5* é uma linguagem estruturante baseada em marcadores e permite apresentar o conteúdo para a WWW. A indicação do "5" indica a quinta versão desta linguagem. Este permitiu a construção das páginas onde o cliente pode navegar e utilizar a aplicação. Foi também utilizado o *localStorage* do *HTML5* para armazenar várias informações necessárias ao funcionamento da aplicação.

Bootstrap

Bootstrap é um conjunto de ferramentas de *frontend* para desenvolvimento rápido de aplicações *web*.

Esta possui modelos de design baseados em HyperText Markup Language (HTML) e Cascading Style Sheets (CSS) para tipografia, formulários, botões, navegação, entre outros componentes de interface. Possui também extensões em JavaScript para funcionalidades adicionais como *popups*, animações nas transições, entre outros.

Assim o *Bootstrap* fornece uma solução limpa e uniforme para a criação de *interfaces web* criando assim protótipos simples, rápidos e altamente utilizáveis.

JavaScript

O *JavaScript* é uma linguagem de programação do lado do cliente. É uma das principais linguagens *client-side* nos *browsers*. Esta permitiu definir as várias funções da interação do cliente com a aplicação no protótipo desenvolvido.

jQuery

O *jQuery* é uma biblioteca do JavaScript *cross-browser* e permite auxiliar a criação de *scripts* do lado do cliente que interagem com o HTML. Foi utilizado para manipulação e controlo dos dados inseridos pelos utilizadores em conjugação com *JavaScript*.

4.6.3 Node-RED

Node-RED é uma ferramenta de programação baseada em fluxos, originalmente desenvolvida pela equipa da *IBM Emerging Technology* e desde 2016 é mantida pela *JS Foundation* como um projeto *Open Source*.

Esta ferramenta multi-plataforma é compatível com diversos microcontroladores conhecidos como o *Arduino*, *Raspberry Pi*, *Beaglebone* e *Intel Edison* com o intuito de criar dispositivos para a IoT. Foi desenvolvido em *Node.js* e é possível configurar e personalizar através da sua interface gráfica.

O *Node-RED* permite comunicar com os microcontroladores, através de protocolos já conhecidos como o HTTP, MQTT, TCP e UDP, para além disso é possível interagir com redes sociais e email.

Através do seu editor é possível adicionar e configurar vários nós e interligá-los criando assim um fluxo dos dados.

O conjunto de nós pode ser facilmente estendido através da instalação de *plugins* muitas vezes criados pela própria comunidade.

É possível importar e exportar os fluxos criados através de documentos JSON que poderão ser compartilhados com a comunidade.

4.6.4 Home-Assistant

Home Assistant é uma plataforma *Open Source* de domótica desenvolvida em *Python 3*. Através desta plataforma é possível controlar e verificar todos os equipamentos em casa. O objetivo do *Home Assistant* é colmatar algumas dificuldades atualmente na domótica, onde

muitas empresas com os seus sistemas proprietários não permitem a interoperabilidade entre os vários dispositivos. Por outro lado, as próprias comunicações realizando-se através de diversos protocolos dificultam a sua integração. Com o *Home Assistant*, tudo isso deixou de ser um problema.

A *interface* é acessível e multi-plataforma permitindo a instalação nos vários sistemas operativos, tal como *Windows*, *Mac* e *Linux*.

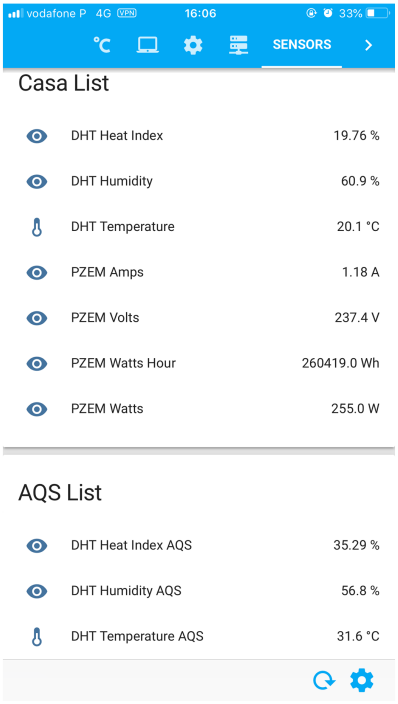

Figura 4.11: Home-Assistant

Através da sua aplicação é possível perceber o funcionamento do seu ecossistema onde podemos adicionar vários componente e definir regras de funcionamento, como por exemplo: acender uma lâmpada especifica após a atuação de um sensor (Figura 4.11).

4.7 Benchmarks

Sendo esta implementação uma plataforma IoT, o desempenho é de grande importância, pois irá lidar com uma grande quantidade de dispositivos e solicitações de serviço. Será testada nesta secção a API da implementação da arquitetura anteriormente apresentada em termos de desempenho, com especial foco no impacto que a segurança cria. Na Figura 4.12 pode ser visto um pedido realizado à API utilizando a autenticação necessária à sua utilização.

Foram utilizadas duas máquinas para a implementação e suporte das múltiplas soluções necessárias para um caso de uso completo. A Tabela 4.2 apresenta as especificações desses computadores.

Os testes realizados foram criados utilizando *mocha*. *Mocha* é uma *framework* de testes *JavaScript* para a linguagem de programação *Node.js*. Os testes *mocha* são executados em série,

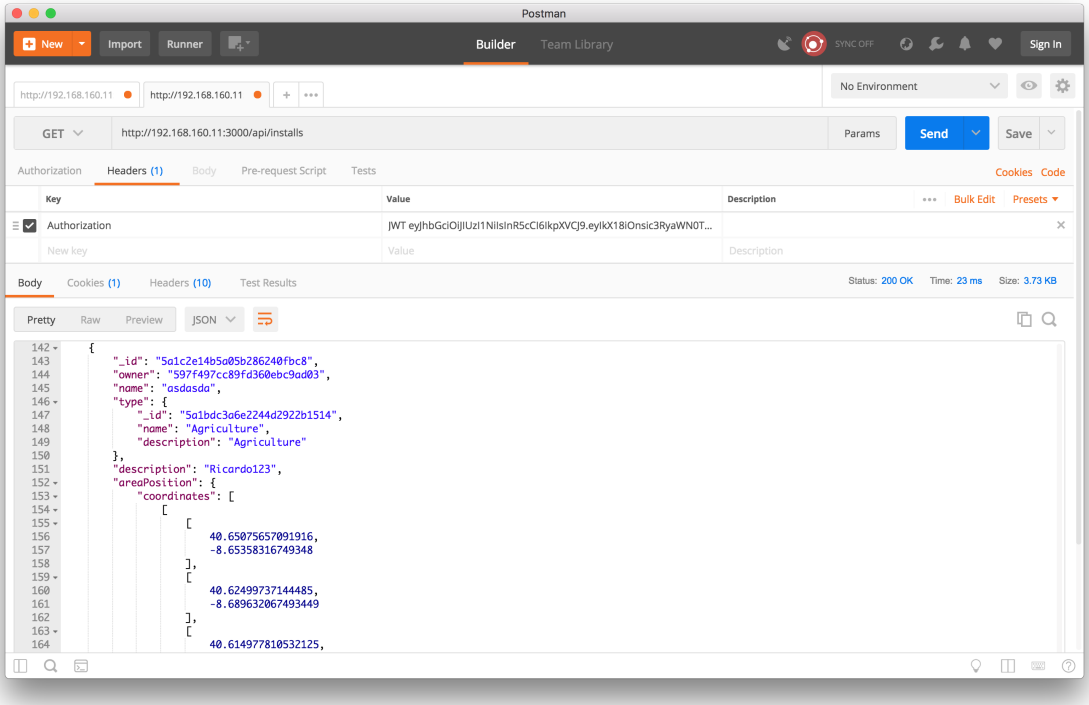

Figura 4.12: Exemplo de utilização da API

permitindo criar relatórios dos mesmos. Para tal a execução dos testes basta simplesmente executar *npm test*.

#	máquina	Sistema Operativo	CPU	RAM
1	MacBook Pro (Retina, Mid 2012)	macOS 10.13.6	Intel Core i7 2,3 GHz	16 GB 1600 MHz DDR3
2	máquina Virtual	Ubuntu Server 16.04.3	CPU-1 VCPU-1	4 GB

Tabela 4.2: Especificação das máquinas de testes

O objetivo dos testes à API é permitir de uma forma rápida e precisa verificar o correcto funcionamento da mesma. Também é possível verificar, mesmo que de uma forma superficial, o desempenho e disponibilidade da API como pode ser constatado a partir das Figuras 4.13 e 4.14.

Para concluir, a métrica com maior peso no desempenho do sistema, é a autenticação. Portanto, para melhorar o desempenho, as otimizações podem ser feitas nos mecanismos de segurança. O design flexível da plataforma, também permite adicionar diferentes mecanismos de segurança, por exemplo, permitir uma segurança simplificada para casos de uso em que a segurança não seja uma prioridade.

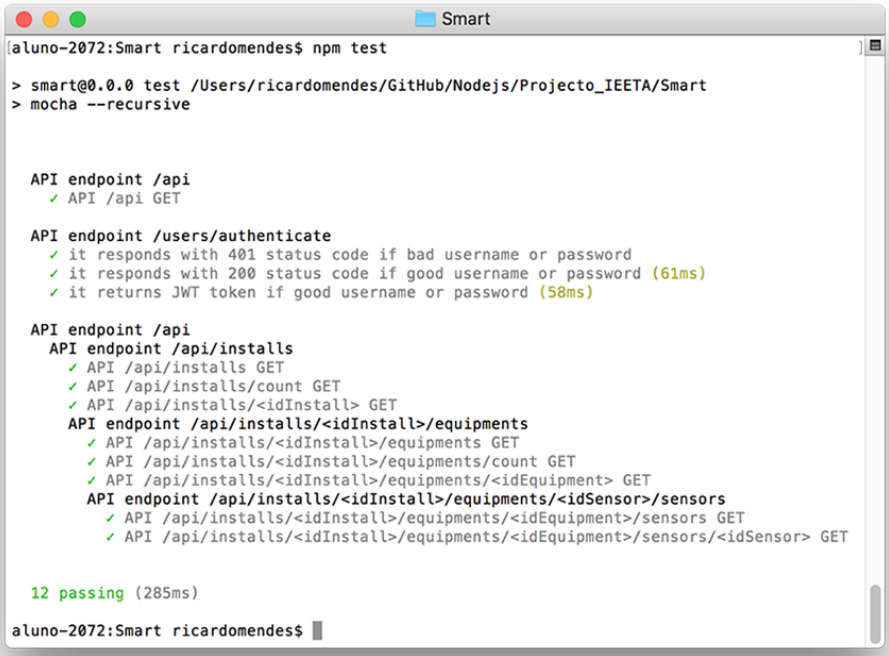

Figura 4.13: Testes API - Local

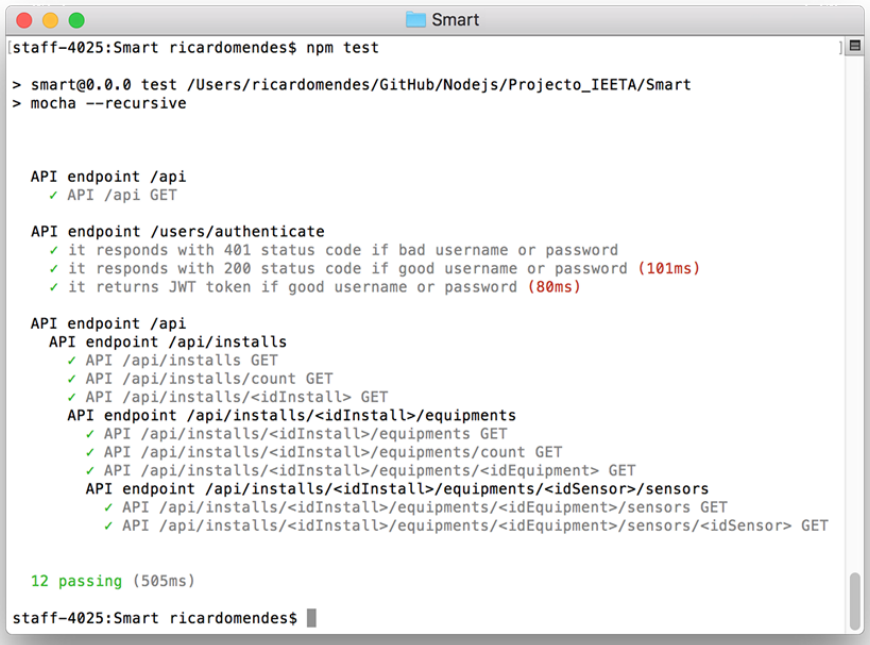

Figura 4.14: Testes API - Virtual

CAPÍTULO 5

Conclusões

Com o surgimento das WSN e restantes redes de sensores tornou-se possível a monitorização remota de um enorme conjunto de domínios, sejam eles cuidados de saúde, fitness, recuperação de lesões, bem como monitorização de uma colheita na agricultura e acompanhamento de índices de poluição. Todos estes cenários são capazes de gerar uma quantidade enorme de dados que terão de ser transmitidos e analisados. É sem dúvida um desafio de *Big Data* que exige uma abordagem escalável para armazenamento, processamento e posterior análise.

A arquitetura apresentada fornece uma plataforma para armazenamento, tratamento, processamento e apresentação individualizada de dados provenientes de redes de sensores. Assim sendo é um sistema escalável e flexível capaz de gerir uma multiplicidade de sensores permitindo assim uma divulgação de dados e informação através da rede. Desta forma é possível que equipas multidisciplinares consigam monitorizar parâmetros específicos dos seus âmbitos sobre os mesmos dados. Os dados recolhidos podem ser disponibilizados em tempo real para as diversas equipas bem como armazenados e disponibilizados para posteriores analises. Assim, os dados poderão ser acedidos em qualquer momento.

Sem dúvida que numa fase seguinte a segurança terá de ter um papel ainda mais importante. Muitas destas redes de sensores lidam com dados de pacientes, atletas, mas também dados de uma produção hidropónica com um nível de confidencialidade elevado e que não deverão estar acessíveis a terceiros. Porém, as maiores limitações ao nível de segurança encontram-se nos dispositivos que ainda não têm capacidade de processamento suficiente para lidarem com técnicas de segurança mais avançadas.

Por outro lado, dependendo do âmbito do projeto, poderão ser adicionados outros sensores que permitam a análise de pH, turvação da água, sensores de pulsação, entre outros, sem que seja necessário alterar a arquitetura.

Nesse sentido, podemos identificar que é possível evoluir. Tratando-se de uma arquitetura escalável e expansível através de *plugins*, dependendo do âmbito, poderá haver necessidade de um tratamento mais rigoroso e seguro dos dados, exigindo assim um processo de aplicação dos conceitos aqui apresentados.

Torna-se assim, evidente que é necessária toda uma camada de segurança na comunicação dentro e fora da arquitetura. Será necessário garantir que os dados não são acessíveis fora da aplicação, garantindo assim total segurança no armazenamento e manuseamento da informação.

Para além das questões de segurança, existem muitos outros fatores que poderão ser melhorados neste projeto. Alguns dos desafios de melhoria incluem a possibilidade de acesso a dados *offline*, possibilitando o acesso direto aos dados lidos através da rede de sensores. Muitas vezes, as redes de sensores encontram-se distantes dos meios urbanos o que dificulta a ligação desses mesmos sensores a infraestruturas que permitam a ligação dos sensores aos restantes módulos.

Por outro lado, a possibilidade de configuração remota dos vários nós que formam a rede de sensores, através da infraestrutura seria uma mais valia pois não haveria a necessidade de uma deslocação no caso de uma atualização do sistema, por exemplo.

Outro desafio interessante a acrescentar é a possibilidade de adição de novos *plugins* ao *Flow Manager* que permita a manipulação dos dados através de *machine learning* utilizando para tal a arquitetura desenvolvida.

Muitos são os desafios que podem assentar em cima desta proposta de arquitetura. Trata-se de uma área que, apesar de todos os avanços que têm sido feitos, é sempre possível levar mais além e com isso facilitar e melhorar a vida das pessoas aproximando-as deste mercado emergente da IoT.

Referências

[1] IBM, P. Zikopoulos e C. Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, 1st. McGraw-Hill Osborne Media, 2011, ISBN: 0071790535, 9780071790536.

[2] J. Yick, B. Mukherjee e D. Ghosal, «Wireless sensor network survey», *Computer Networks*, vol. 52, nº 12, pp. 2292–2330, 2008, ISSN: 13891286. DOI: 10.1016/j.comnet.2008.04.002. arXiv: 1011.1529.

[3] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao e V. C. M. Leung, *Body area networks: A survey*, 2011. DOI: 10.1007/s11036-010-0260-8.

[4] G. Fortino, G. Di Fatta, M. Pathan e A. V. Vasilakos, «Cloud-assisted body area networks: state-of-the-art and future challenges», *Wireless Networks*, vol. 20, nº 7, pp. 1925–1938, 2014, ISSN: 10220038. DOI: 10.1007/s11276-014-0714-1.

[5] a. Zanella, N. Bui, a. Castellani, L. Vangelista e M. Zorzi, «Internet of Things for Smart Cities», *IEEE Internet of Things Journal*, vol. 1, nº 1, pp. 22–32, 2014, ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2306328. arXiv: arXiv:1011.1669v3. endereço: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6740844.

[6] G. Fortino, D. Parisi, V. Pirrone e G. Di Fatta, «BodyCloud: A SaaS approach for community Body Sensor Networks», *Future Generation Computer Systems*, vol. 35, pp. 62–79, 2014, ISSN: 0167739X. DOI: 10.1016/j.future.2013.12.015.

[7] D. Evans, «The Internet of Things - How the Next Evolution of the Internet is Changing Everything», *CISCO white paper*, nº April, pp. 1–11, 2011, ISSN: 09598138. DOI: 10.1109/IEEESTD.2007.373646. arXiv: arXiv:1011.1669v3. endereço: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:The+Internet+of+Things+-+How+the+Next+Evolution+of+the+Internet+is+Changing+Everything%7B%5C#%7D0.

[8] J. Rivera e R. Van der Muelen, *Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020*, 2013. endereço: http://www.gartner.com/newsroom/id/2636073.

[9] J. Bradley, C. Reberger, A. Dixit e V. Gupta, «Internet of Everything : A $ 4.6 Trillion Public-Sector Opportunity», *Cisco*, nº 23, pp. 1–17, 2013. endereço: https://www.cisco.com/c/dam/en%7B%5C_%7Dus/about/business-insights/docs/ioe-public-sector-vas-white-paper.pdf.

[10] Cisco, *Internet of Things (IoT)*, 2017. endereço: https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html.

[11] N. Hunke, Z. Yusuf, M. Rüßmann, F. Schmieg, A. Bhatia e N. Kalra, *Winning in IoT: It's All About the Business Processes*, 2017. endereço: https://www.bcg.com/publications/2017/hardware-software-energy-environment-winning-in-iot-all-about-winning-processes.aspx.

[12] R. van der Meulen, *Gartner Says 8.4 Billion Connected "Things"Will Be in Use in 2017, Up 31 Percent From 2016*, 2017. endereço: http://www.gartner.com/newsroom/id/3598917.

[13] Harbor Research, *Internet of Things Infographic | What Is The "Internet of Things"?*, 2016. endereço: http://www.postscapes.com/what-exactly-is-the-internet-of-things-infographic/.

[14] B. Antonescu e S. Basagni, «Wireless Body Area Networks: Challenges, Trends and Emerging Technologies», *Proceedings of the 8th International Conference on Body Area Networks*, pp. 1–7, 2013. DOI: 10.4108/icst.bodynets.2013.253722. endereço: http://dl.acm.org/citation.cfm?id=2555319.2555321.

[15] S. May, *Engineering Design Process*, 2018. endereço: https://www.nasa.gov/audience/foreducators/best/edp.html.

[16] S. Geisler, «Data Stream Management Systems», *Dagstuhl Follow-Ups*, vol. 5, pp. 275–304, 2013, ISSN: 1868-8977. DOI: 10.4230/DFU.Vol5.10452.275. endereço: http://drops.dagstuhl.de/opus/volltexte/2013/4297/.

[17] Q. Wang e I. Balasingham, *Wireless Sensor Networks - An Introduction*. INTECH Open Access Publisher, 2010, ISBN: 9789533073217. endereço: https://books.google.pt/books?id=rkCRoAEACAAJ.

[18] C.-y. Chong, S. P. Kumar e S. Member, «Sensor Networks : Evolution , Opportunities , and Challenges», vol. 91, nº 8, 2003.

[19] D. C. Steere, A. Baptista, D. Mcnamee, C. Pu e J. Walpole, «Research Challenges in Environmental Observation and Forecasting Systems», pp. 292–299, 2000.

[20] K. Martinez, P. Padhy e A. Riddoch, «Glacial Environment Monitoring using Sensor Networks», 2005.

[21] J. Holler, V. Tsiatsis, C. Mulligan, S. Avesand, S. Karnouskos e D. Boyle, *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. 2014, p. 352, ISBN: 9780080994017. DOI: B978-0-12-407684-6.00001-2. endereço: http://store.elsevier.com/product.jsp?isbn=9780124076846.

[22] A. Qureshi, W. P. Kang, J. L. Davidson e Y. Gurbuz, *Review on carbon-derived, solid-state, micro and nano sensors for electrochemical sensing applications*, 2009. DOI: 10.1016/j.diamond.2009.09.008.

[23] IBM, *IBM Watson*. endereço: https://www.ibm.com/watson/.

[24] AT&T, «What you need to know about IoT», *AT&T Whitepapers*, 2016. endereço: http://www.business.att.com/content/whitepaper/what-you-need-to-know-about-IoT.pdf.

[25] J. Hawn, *In-depth: Top 10 Internet of Things companies to watch*, 2015. endereço: https://www.rcrwireless.com/20151130/internet-of-things/in-depth-top-10-internet-of-things-companies-to-watch.

[26] A. Whitmore, A. Agarwal e L. Da Xu, «The Internet of Things—A survey of topics and trends», *Information Systems Frontiers*, vol. 17, nº 2, pp. 261–274, 2015, ISSN: 15729419. DOI: 10.1007/s10796-014-9489-2.

[27] G. Trickey, «GSMA: Driving Innovation in Connected Living», 2014. endereço: http://www.gsma.com/newsroom/wp-content/uploads/us-m2m-2014.pdf.

[28] G. Intelligence, «From concept to delivery: the M2M market today», *White Paper, Feb*, nº February, pp. 1–21, 2014.

[29] Google, *Google Trends*, 2018. endereço: https://trends.google.pt/trends/.

[30] SmartCitiesWorld, *Smart cities services worth $225bn by 2026*, 2017. endereço: https://www.smartcitiesworld.net/news/news/smart-cities-services-worth-225bn-by-2026-1618.

[31] FIWARE, *PORTO, A CITY THAT HAS BECOME A REAL-TIME GUIDE*, 2015. endereço: https://www.fiware.org/2015/11/20/porto-a-city-that-has-become-a-real-time-guide/.

[32] S. Battle e B. Gaster, «LoRaWAN Bristol», 2017. DOI: 10.1145/3105831.3105835.

[33] S. Al-Sarawi, M. Anbar, K. Alieyan e M. Alzubaidi, «Internet of Things (IoT) communication protocols: Review», 2017. DOI: 10.1109/ICITECH.2017.8079928.

[34] R. Ayers, *How Internet of Things will change the Sharing Economy*, 2017. endereço: https://bigdata-madesimple.com/how-internet-of-things-will-change-the-sharing-economy/.

[35] E. Kim, D. Kaspar e J. P. Vasseur, *Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, RFC 6568, 2012. DOI: 10.17487/RFC6568. endereço: https://rfc-editor.org/rfc/rfc6568.txt.

[36] Sigfox, *Sigfox adding 4 new countries and set to cover more than 80% of Latin American population*, 2018. endereço: https://www.sigfox.com/en/news/sigfox-adding-4-new-countries-and-set-cover-more-80-latin-american-population.

[37] S. Greengard, *The Internet of Things*, sér. The MIT Press Essential Knowledge series. MIT Press, 2015, ISBN: 9780262527736. endereço: https://books.google.pt/books?id=oyyyBwAAQBAJ.

[38] David S. Watson, Mary Ann Piette, Osman Sezgen e Naoya Motegi, «Machine to Machine (M2M) Technology in Demand Responsive Commercial Buildings», *2004 ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA*, nº August, 2004. DOI: 10.1021/acs.nanolett.7b00249.

[39] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach e T. Berners-Lee, «RFC2616 - Hypertext transfer protocol–HTTP/1.1», *Internet Engineering Task Force*, pp. 1–114, 1999, ISSN: 20701721. DOI: http://www.ietf.org/rfc/rfc2616.txt. endereço: http://tools.ietf.org/html/rfc2616.

[40] T. Levä, O. Mazhelis e H. Suomi, «Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications», *Decision Support Systems*, vol. 63, nº Supplement C, pp. 23–38, 2014, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2013.09.009. endereço: http://www.sciencedirect.com/science/article/pii/S0167923613002376.

[41] R. Gravina, C. Ma, P. Pace, G. Aloi, W. Russo, W. Li e G. Fortino, «Cloud-based Activity-aaService cyber–physical framework for human activity monitoring in mobility», *Future Generation Computer Systems*, vol. 75, pp. 158–171, 2017, ISSN: 0167739X. DOI: 10.1016/j.future.2016.09.006. endereço: http://dx.doi.org/10.1016/j.future.2016.09.006.

[42] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski e R. Jafari, «Enabling effective programming and flexible management of efficient body sensor network applications», *IEEE Transactions on Human-Machine Systems*, vol. 43, nº 1, pp. 115–133, 2013, ISSN: 21682291. DOI: 10.1109/TSMCC.2012.2215852.

[43] D. Malan e T. Fulford-Jones, «Codeblue: An ad hoc sensor network infrastructure for emergency medical care», *... Workshop on Wearable ...*, pp. 3–6, 2004. endereço: http://icawww.epfl.ch/luo/WAMES%202004%7B%5C_%7Dfiles/WAMESproceedings.pdf%7B%5C#%7Dpage=12.

[44] P. Kuryloski, A. Giani, R. Giannantonio, K. Gilani, R. Gravina, V.-P. Seppa, E. Seto, V. Shia, C. Wang, P. Yan, A. Y. Yang, J. Hyttinen, S. Sastry, S. Wicker e R. Bajcsy, «DexterNet: An Open Platform for Heterogeneous Body Sensor Networks and its Applications», em *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, 2009, pp. 92–97, ISBN: 978-0-7695-3644-6. DOI: 10.1109/BSN.2009.31. endereço: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5226911%7B%5C%%7D5Cnhttps://docs.google.com/file/d/0ByzBYZoteKyfOVlkQ2lRbFhPbUE/edit.

[45] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi, R. Jafari e G. Fortino, «From modeling to implementation of virtual sensors in body sensor networks», *IEEE Sensors Journal*, vol. 12, nº 3, pp. 583–593, 2012, ISSN: 1530437X. DOI: 10.1109/JSEN.2011.2121059.

[46] S. Madria, V. Kumar e R. Dalvi, «Sensor cloud: A cloud of virtual sensors», *IEEE Software*, vol. 31, nº 2, pp. 70–77, 2014, ISSN: 07407459. DOI: 10.1109/MS.2013.141. arXiv: arXiv:0906.1623v1.

[47] M. Bondarenko, *Industrial IoT Innovation*, 2018. endereço: https://www.kaaproject.org/industrial-iot-innovation/.

[48] Eclipse, *Ponte by Eclipse*, 2018. endereço: http://www.eclipse.org/ponte/.

[49] OpenMTC, *OpenMTC*, 2018. endereço: https://www.openmtc.org.

[50] Altair, *How Altair SmartCore Works*, 2018. endereço: https://www.altairsmartworks.com/smartcore-howitworks.

[51] LogMeIn, *Xively by LogMeIn*, 2018. endereço: https://www.xively.com/xively-iot-platform.

[52] Ayantek, *A Comparative Analysis of IoT Platforms for the Medical Devices Industry*, 2015. endereço: http://www.ayantek.com/a-comparative-analysis-of-iot-platforms-for-the-medical-devices-industry/.

[53] Libelium, *Waspmote*, 2017. endereço: http://www.libelium.com/products/waspmote/.

[54] Espressif, *ESP8266*, 2017. endereço: `https://www.espressif.com` (acedido em 2017).

[55] M. Collina, *Mosca*, 2017. endereço: `http://www.mosca.io`.

[56] IBM, *MQTT*, 2017. endereço: `http://mqtt.org`.

[57] Jared Hanson, *Passportjs*, 2017. endereço: `http://www.passportjs.org`.

[58] JS Foundation, *Node-red*, 2017. endereço: `http://nodered.org`.

[59] J. Lehnardt, *mustache.js*, 2018. endereço: `https://github.com/janl/mustache.js`.

[60] M. Bostock, *TopoJSON*. endereço: `https://github.com/topojson/topojson/wiki`.

Anexo A

Serviços

Ao longo deste anexo é pretendido apresentar os serviços necessários ao bom funcionamento do protótipo desenvolvido.

Inicialmente será apresentado no Código 1 o serviço responsável pela base de dados *mongoDB*.

```
[Unit]
Description=High-performance, schema-free document-oriented database
After=network.target
Documentation=https://docs.mongodb.org/manual

[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/bin/mongod --config /etc/mongod.conf
PIDFile=/var/run/mongodb/mongod.pid
# file size
LimitFSIZE=infinity
# cpu time
LimitCPU=infinity
# virtual memory size
LimitAS=infinity
# open files
LimitNOFILE=64000
# processes/threads
LimitNPROC=64000
# locked memory
LimitMEMLOCK=infinity
# total threads (user+kernel)
TasksMax=infinity
TasksAccounting=false

[Install]
WantedBy=multi-user.target
```

Código 1: mongod.service

De seguida será apresentado o serviço responsável pelo servidor MQTT no Código 2.

```
[Unit]
Description=Start mosca node.js app

[Service]
WorkingDirectory=/home/oneadmin/mosca
ExecStart=/usr/bin/node /home/oneadmin/mosca/index.js
Restart=always
```

Código 2: mosca.service

Por fim será apresentado o serviço responsável pelo servidor do protótipo desenvolvido que poderá ser analisado no Código 3.

```
[Unit]
Description=Start Smart node.js app

[Service]
WorkingDirectory=/home/oneadmin/Smart
ExecStart=/usr/bin/node /home/oneadmin/Smart/bin/www
Restart=always
```

Código 3: smart.service

Estes são serviços fundamentais para o bom funcionamento de todo o protótipo.

Anexo B

Modelo de Dados

Neste anexo serão descrito os modelos de dados elaborados para suporte à arquitetura.

Modelo de Dados de suporte à configuração dos módulos

User

O modelo *User* tem como objetivo armazenar a informação referente aos utilizadores da plataforma.

```
mongoose.Schema({
        local: {
        username: String,
        password: String
    },
    facebook: {
        id: String,
        token: String,
        email: String,
        name: String
    },
    twitter: {
        id: String,
        token: String,
        displayName: String,
        username: String
    },
    google: {
        id: String,
        token: String,
        email: String,
        name: String
    },

    urlNodeRed: String,
    updateFlowDate: Date,
    pid: String
});
```

Código 4: User model

A Tabela 1 apresenta uma explicação dos campos do modelo *User*.

Atributo	Descrição
local.username	Utilizador local
local.password	*Password* do utilizador local
facebook.id	Identificador da conta do *Facebook*
facebook.token	*Token* da conta do *Facebook*
facebook.email	Email associado à conta do *Facebook*
facebook.name	Nome associado à conta do *Facebook*
twitter.id	Identificador da conta do *Twitter*
twitter.token	*Token* da conta do *Twitter*
twitter.displayName	Nome de apresentação associado à conta do *Twitter*
twitter.username	Nome associado à conta do *Twitter*
google.id	Identificador da conta do *Google*
google.token	*Token* da conta do *Google*
google.email	Email associado à conta do *Google*
google.name	Nome associado à conta do *Google*
urlNodeRed	Endereço do *Flow Manager*
updateFlowDate	Data da última atualização do *Flow Manager*
pid	Número do processo do *Flow Manager* do utilizador

Tabela 1: User Model

Install

O modelo *Install* tem como objetivo armazenar a informação referente às instalações de um utilizador.

```
mongoose.Schema({
        owner: {type: mongoose.Schema.Types.ObjectId, ref: 'user'},
    name: String,
    type: {type: mongoose.Schema.Types.ObjectId, ref: 'type'},
    description: String,
    position: {
        type: [Number],  // [<longitude>, <latitude>]
        index: '2d'      // create the geospatial index
    },
    areaPosition: mongoose.Schema.Types.Polygon
});
```

Código 5: Install model

A Tabela 2 apresenta uma explicação dos campos do modelo *Install*.

Atributo	Descrição
owner	Identificador do utilizador
name	Nome da Instalação
type	Identificador do tipo de instalação
description	Descrição da Instalação
position	Localização geoespacial da Instalação
areaPosition	Representação da área da Instalação

Tabela 2: Install Model

Type

O modelo *Type* tem como objetivo armazenar os vários tipos de instalação.

```
mongoose.Schema({
        name: String,
    description: String
});
```

Código 6: Type model

A Tabela 3 apresenta uma explicação dos campos do modelo *Type*.

Atributo	**Descrição**
name	Nome do tipo de Instalação
description	Descrição do tipo de Instalação

Tabela 3: Type Model

Equipment

O modelo *Equipment* tem como objetivo armazenar os equipamentos de uma instalação.

```
mongoose.Schema({
        owner: {type: mongoose.Schema.Types.ObjectId, ref: 'user'},
    install: {type: mongoose.Schema.Types.ObjectId, ref: 'install'},
    name: String,
    description: String,
    position: {
        type: [Number],  // [<longitude>, <latitude>]
        index: '2d'      // create the geospatial index
    }
});
```

Código 7: Equipment model

A Tabela 4 apresenta uma explicação dos campos do modelo *Equipment*.

Atributo	**Descrição**
owner	Identificador do utilizador
install	Identificador da Instalação respetivo
name	Nome do Equipamento
description	Descrição do Equipamento
position	Localização geoespacial do Equipamento

Tabela 4: Equipment Model

Sensor

O modelo *Sensor* tem como objetivo armazenar os sensores de um equipamento.

A Tabela 5 apresenta uma explicação dos campos do modelo *Sensor*.

```
mongoose.Schema({
         owner: {type: mongoose.Schema.Types.ObjectId, ref: 'user'},
    install: {type: mongoose.Schema.Types.ObjectId, ref: 'install'},
    equipment: {type: mongoose.Schema.Types.ObjectId, ref: 'equipment'},
    name: String,
    description: String,
        dataType: {type: mongoose.Schema.Types.ObjectId, ref: 'dataType'},
    chartType: {type: mongoose.Schema.Types.ObjectId, ref: 'chartType'}
});
```

Código 8: Sensor model

Atributo	Descrição
owner	Identificador do utilizador
install	Identificador da Instalação respetiva
equipment	Identificador do Equipamento respetivo
name	Nome do Sensor
description	Descrição do Sensor
dataType	Identificador do tipo de dados
chartType	Identificador do tipo de gráfico

Tabela 5: Sensor Model

DataType

O modelo *DataType* tem como objetivo armazenar os vários tipos de dados.

```
var dataTypeSchema = mongoose.Schema({
        name: String,
    description: String
});
```

Código 9: DataType model

A Tabela 6 apresenta uma explicação dos campos do modelo *DataType*.

Atributo	Descrição
name	Nome do tipo de dados
description	Descrição do tipo de dados

Tabela 6: Data Type Model

ChartType

O modelo *ChartType* tem como objetivo armazenar os vários tipos de gráficos.

A Tabela 7 apresenta uma explicação dos campos do modelo *ChartType*.

Atributo	Descrição
name	Nome do tipo de gráfico
description	Descrição do tipo de gráfico

Tabela 7: Chart Type Model

```
var chartTypeSchema = mongoose.Schema({
        name: String,
    description: String
});
```

Código 10: ChartType model

Anexo C

Flow Manager - Instâncias

Este anexo pretende deixar uma memória futura para a criação de uma instâncias do *Flow Manager* baseado em *Node-RED*, permitindo assim, caso necessário, a criação de novas instâncias do *Flow Manager* no futuro por cada instalação, sendo que no protótipo desenvolvido estes passos são gerados dinamicamente.

Criação de novas instância

Para criação de uma nova instância do módulo *Flow Manager* é necessário a configuração de dois ficheiros:

- settings.js
- flow.json

No ficheiro *settings.js* estão presentes as configurações básicas para a criação de uma nova instância. Dessas configurações podemos destacar os seguintes parâmetros: *uiPort*, *flowFile*, *userDir*, *adminAuth* e *editorTheme*.

```
module.exports = {
    uiPort: {{uiPort}},
    mqttReconnectTime: 15000,
    flowFile: 'flows/flow_{{idUser}}.json',
    userDir: '{{{userDir}}}',
    httpRoot: '/red',
    ui: {path: "ui"},
    adminAuth: require("../user-authentication"),
    editorTheme: {
        page: {
            title: "Smart*", favicon: "editor/favicon.ico",
        },
        header: {
            title: "Smart*",
        },
        menu: {
            "menu-item-import-library": false, "menu-item-import":false,
            "menu-item-keyboard-shortcuts": false, "menu-item-search":false,
            "menu-item-edit-palette":false, "menu-item-show-tips":false
        },
    }
}
```

Código 11: Template settings.js

O parâmetro *uiPort* permite definir o porto TCP a utilizar. No *flowFile* é possível definir a localização do ficheiro *flow.json* bem como no *userDir* definir a localização dos restantes ficheiros associados à nova instância. Com o parâmetro *adminAuth* define-se o método de autenticação na nova instância. Por fim, através do parâmetro *editorTheme* é possível personalizar a visualização do *Flow Manager*.

A fim de facilitar a criação de novas instâncias dinamicamente foi utilizado o módulo *mustache.js*, onde os paramentos são preenchidos automaticamente a partir dos dados presentes no módulos de *Authentication* e *System, Device, Sensor Registration*.

No Código 11 será apresentado o template que permite a criação do ficheiro *settings.js*. No final da criação deste ficheiro é executado *node red.js –settings settings.js* para finalizar o processo.

```
[
  {
    "id": "8bec196c.723a18",
    "type": "mqtt-broker",
    "z": "",
    "broker": "{{ipBroker}}",
    "port": "{{portBoker}}",
    "clientid": "",
    "usetls": false,
    "compatmode": true,
    "keepalive": "60",
    "cleansession": true,
    "willTopic": "",
    "willQos": "0",
    "willPayload": "",
    "birthTopic": "",
    "birthQos": "0",
    "birthPayload": ""
  }
]
```

Código 12: Template nodeRed_flow_template_broker.json

Para que seja criado uma visualização é necessário o ficheiro *flow.json*. No Código 12 está presente o template utilizado para a criação de um *broker* dentro do *flow.json*. Par tal é possível definir os parâmetros *ipBroker* e *portBoker*.

O Código 13 permite demonstrar a criação de um interruptor para adição ao ficheiro *flow.json*. Os seguintes paramentos podem ser definidos no template:*sensorInstall*, *sensorEquipment*, *sensorID*, *sensorName* e *sensorType*.

Por fim, utilizado o módulo *mustache.js* o ficheiro *flow.json* é criado e utilizado pelo *Flow Manager*.

```
[
  {
    "id": "{{sensorID}}_read",
    "type": "mqtt in",
    "z": "{{sensorInstall}}",
    "name": "{{sensorName}}_input",
    "topic": "{{sensorInstall}}/{{sensorEquipment}}/{{sensorID}}/1/read",
    "qos": "2",
    "broker": "8bec196c.723a18",
    "x": {{posX1}}, "y": {{posY}},
    "wires": [ ["{{sensorID}}_function"] ]
  }, {
    "id": "{{sensorID}}_function",
    "type": "function",
    "z": "{{sensorInstall}}",
    "name": "{{sensorName}}_function",
    "func": "var json = JSON.parse(msg[\"payload\"]);\n\nvar result= json[\"{{sensorType}}\"];",
    "outputs": 1,
    "noerr": 0,
    "x": {{posX2}}, "y": {{posY}},
    "wires": [ ["{{sensorID}}_write"] ]
  }, {
    "id": "{{sensorID}}_write",
    "type": "ui_switch",
    "z": "{{sensorInstall}}",
    "name": "{{sensorName}}_write",
    "group": "{{sensorEquipment}}_group",
    "passthru": true,
    "decouple": "false",
    "onvalue": "1",
    "onvalueType": "num",
    "onicon": "", "oncolor": "",
    "offvalue": "0",
    "offvalueType": "num",
    "officon": "",
    "offcolor": "",
    "label": "{{sensorName}}",
    "chartType": "line",
    "interpolate": "linear",
    "removeOlder": 1,
    "removeOlderUnit": "60",
    "x": {{posX3}}, "y": {{posY}},
    "wires": [ ["{{sensorID}}_relay"] ]
  }, {
    "id": "{{sensorID}}_relay",
    "type": "{{sensorType}}",
    "z": "{{sensorInstall}}",
    "name": "{{sensorName}}",
    "topic": "{{sensorInstall}}/{{sensorEquipment}}/{{sensorID}}/1/write",
    "broker": "8bec196c.723a18",
    "x": {{posX4}}, "y": {{posY}},
    "wires": []
  }
]
```

Código 13: Template nodeRed_flow_template_switch.json

Anexo D

Flow Manager - Plugins

Este anexo pretende deixar uma memória futura para o desenvolvimento de novos *plugins* a adicionar ao *Flow Manager*, permitindo assim adicionar novas funcionalidades a este módulo.

Criação de um plugin

Inicialmente crie um diretório onde estarão presentes os ficheiros do *plugin*. Dentro desse diretório, crie os seguintes arquivos:

- package.json
- temperatura_ar.js
- temperatura_ar.html

Deve ser criado um ficheiro *package.json* que define o módulo, um ficheiro *JavaScript* que define o que o *plugin* faz e um ficheiro *HTML* que define as propriedades visuais do *plugin*.

Este exemplo mostrará como criar um *plugin* que representa a aquisição de dados a partir de um sensor que permite a leitura da temperatura do ar.

Após a edição do *plugin* basta executar o novo módulo através do comando *npm*.

package.json

Este é um ficheiro padrão usado pelos módulos de *Node.js* para descrever o seu conteúdo.

Para gerar o ficheiro *package.json*, basta executar o comando *npm init*. De seguida serão apresentadas uma série de perguntas a fim de criar o conteúdo inicial. No Código 14 é apresentado a versão de um *plugin* utilizado no protótipo que contem vários nós.

```
{
  "name": "node-red-contrib-precAgriNodes",
  "version": "0.0.1",
  "description": "A agriculture nodes for node-red",
  "appKey": "0",
  "dependencies": {
    "is-utf8": "^0.2.1",
    "mqtt": "^1.13.0"
  },
  "keywords": [
    "node-red"
  ],
  "node-red": {
    "nodes": {
      "remote_server": "remote_server/remote_server.js",

      "temperatura_ar": "temperatura_ar/temperatura_ar.js",
      "temperatura_solo": "temperatura_solo/temperatura_solo.js",
      "humidade_ar": "humidade_ar/humidade_ar.js",
      "humidade_solo": "humidade_solo/humidade_solo.js",

      "radiacaoSolar": "radiacaoSolar/radiacaoSolar.js",

      "pluviosidade": "pluviosidade/pluviosidade.js",
      "direccao_vento": "direccao_vento/direccao_vento.js",
      "velocidade_vento": "velocidade_vento/velocidade_vento.js",

      "led": "led/led.js",
      "ledStrip": "ledStrip/ledStrip.js",
      "motor": "motor/motor.js",
      "relay": "relay/relay.js"
    }
  }
}
```

Código 14: A agriculture nodes for node-red

temperatura_ar.js

A maioria do *plugin* é definido pela função *Temperatura_arNode*, que é chamada sempre que uma nova instância do nó é criada.

A função chama a função *RED.nodes.createNode* para inicializar os recursos compartilhados por todos os nós. Depois disso, estão presentes alguns parâmetros *default* do *plugin* bem como o seu comportamento.

Finalmente, a função *Temperatura_arNode* é registrada através da função *RED.nodes.registerType*.

Caso seja necessária alguma dependência externa, ela deverá ser incluída na seção de dependências do ficheiro *package.json*.

```
module.exports = function (RED) {
  "use strict";
  var mqtt = require("mqtt");
  var util = require("util");
  var isUtf8 = require('is-utf8');
  var config = require('../package.json');

  function Temperatura_arNode(n) {
    RED.nodes.createNode(this, n);

    this.appKey = config.appKey;
    this.waspId = n.waspId;
    this.dataType = "airtemp";

    this.topic = n.topic;
    this.qos = 2;
    this.broker = n.broker;
    this.brokerConn = RED.nodes.getNode(this.broker);

    if (!/^(#$|(\+|[^+#]*)(\/(\+|[^+#]*))*(\/(\+|#|[^+#]*))?$)/.test(this.topic)) {
      return this.warn(RED._("mqtt.errors.invalid-topic"));
    }
    var node = this;
    if (this.brokerConn) {
      this.status({fill: "red", shape: "ring", text: "node-red:common.status.disconnected"});
      if (this.topic) {
        node.brokerConn.register(this);
        this.brokerConn.subscribe(this.topic, this.qos, function (topic, payload, packet) {
          if (isUtf8(payload)) {
            payload = payload.toString();
          }
          var msg = {topic: topic, payload: payload, qos: packet.qos, retain: packet.retain};
          if ((node.brokerConn.broker === "localhost") || (node.brokerConn.broker === "127.0.0.1")) {
            msg._topic = topic;
          }
          node.send(msg);
        }, this.id);
        if (this.brokerConn.connected) {
          node.status({fill: "green", shape: "dot", text: "node-red:common.status.connected"});
        }
      }
      else {
        this.error(RED._("mqtt.errors.not-defined"));
      }
      this.on('close', function (done) {
        if (node.brokerConn) {
          node.brokerConn.unsubscribe(node.topic, node.id);
          node.brokerConn.deregister(node, done);
        }
      });
    } else {
      this.error(RED._("mqtt.errors.missing-config"));
    }
  }
  RED.nodes.registerType("temperatura_ar", Temperatura_arNode);
};
```

Código 15: temperatura ar js

temperatura_ar.html

```
<script type="text/x-red" data-template-name="temperatura_ar">
  <div class="form-row">
    <label for="node-input-topic">
      <i class="fa fa-globe"></i>
      <span data-i18n="common.label.topic"></span>
    </label>
    <input type="text" id="node-input-topic" data-i18n="[placeholder]common.label.topic">
  </div>
  <div class="form-row">
    <label for="node-input-broker">
      <i class="fa fa-globe"></i>
          <span data-i18n="mqtt.label.broker"></span>
    </label>
    <input type="text" id="node-input-broker">
  </div>
  <div class="form-row">
    <label for="node-input-name">
      <i class="fa fa-tag"></i>
          <span data-i18n="common.label.name"></span>
    </label>
    <input type="text" id="node-input-name" data-i18n="[placeholder]common.label.name">
  </div>
</script>

<script type="text/javascript">
  RED.nodes.registerType('temperatura_ar', {
    category: 'Agricultura',
    color: "#E55934",
    defaults: {
      waspId: {value: ""},
      name: {value: ""},
      topic: {
        value: "",
        required: true,
        validate: RED.validators.regex(/^(#$|(\+|[^+#]*)(\/(\+|[^+#]*))*(\/(\+|#|[^+#]*))?$)/)
      },
      broker: {type:"mqtt-broker", required:true}
    },
    inputs: 0,
    outputs: 1,
    icon: "temperatura_ar.png",
    label: function () {
      return this.name || this.topic || "temperatura_ar";
    },
    labelStyle: function () {
      return this.name ? "node_label_italic" : "";
    }
  });
</script>
```

Código 16: temperatura ar html

É no ficheiro *HTML* que é definida a interface do nó e onde as várias características apresentadas no ficheiro *JavaScript* estão presentes.

No final basta executar o comando *npm install <directório>* para instalar o *plugin* no *Flow Manager*.

Podem ser adicionadas mais informações sobre o módulo, bastando para tal adicionar essas informações no *README* associado.

Caso seja desejado, também é possível adicionar ao *plugin* desenvolvido traduções das mensagens apresentadas. Desta forma é possível implementar o *i18n* no *plugin*.

Anexo E

Manual de Utilização

Ao longo deste anexo é pretendido apresentar o manual de utilização na íntegra para que seja possível perceber a organização e principais funcionalidades do protótipo desenvolvido. Na figura 1 pode-se constatar a interface do protótipo nas várias plataformas. Antes de iniciar a explicação do protótipo desenvolvido é fundamental garantir que os vários serviços estão em execução.

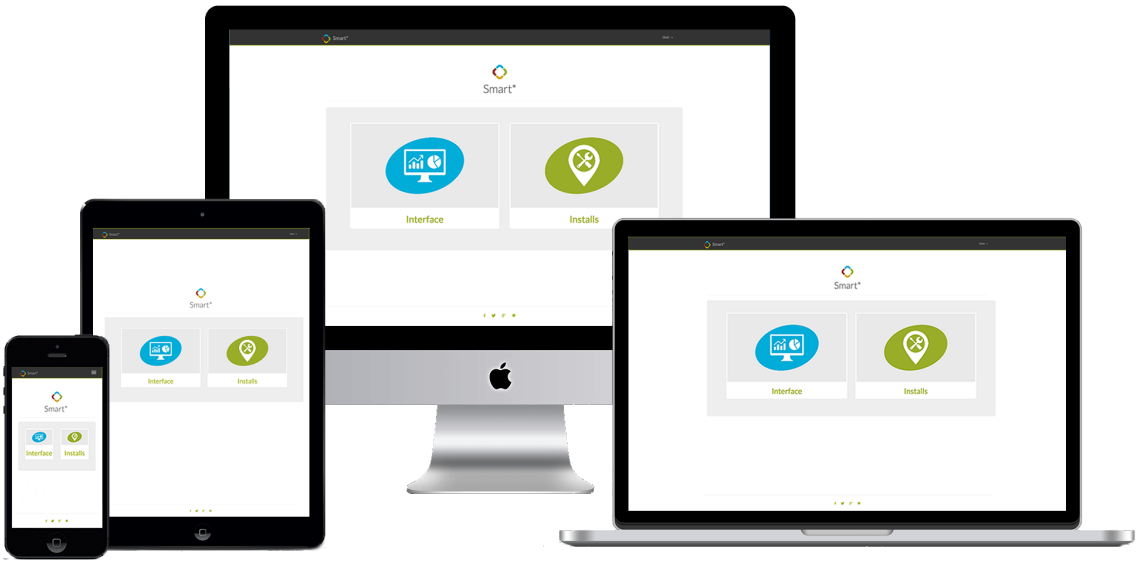

Figura 1: Utilização do protótipo nas diversas plataformas

Protótipo Desktop

As imagens seguintes têm por objetivo demonstrar as funcionalidades do sistema tendo em conta a proposta de arquitetura apresentada.

Quando a aplicação é iniciada, é mostrado ao utilizador um ecrã inicial (Figura 2) onde é possível iniciar a aplicação realizando a sua autenticação através das redes sociais para além da autenticação utilizando o par utilizador/*password* (Figura 3).

Aqui, o utilizador insere as credenciais de acesso e após verificação das mesmas, o utilizador terá acesso então às funcionalidades da aplicação (Figura 4).

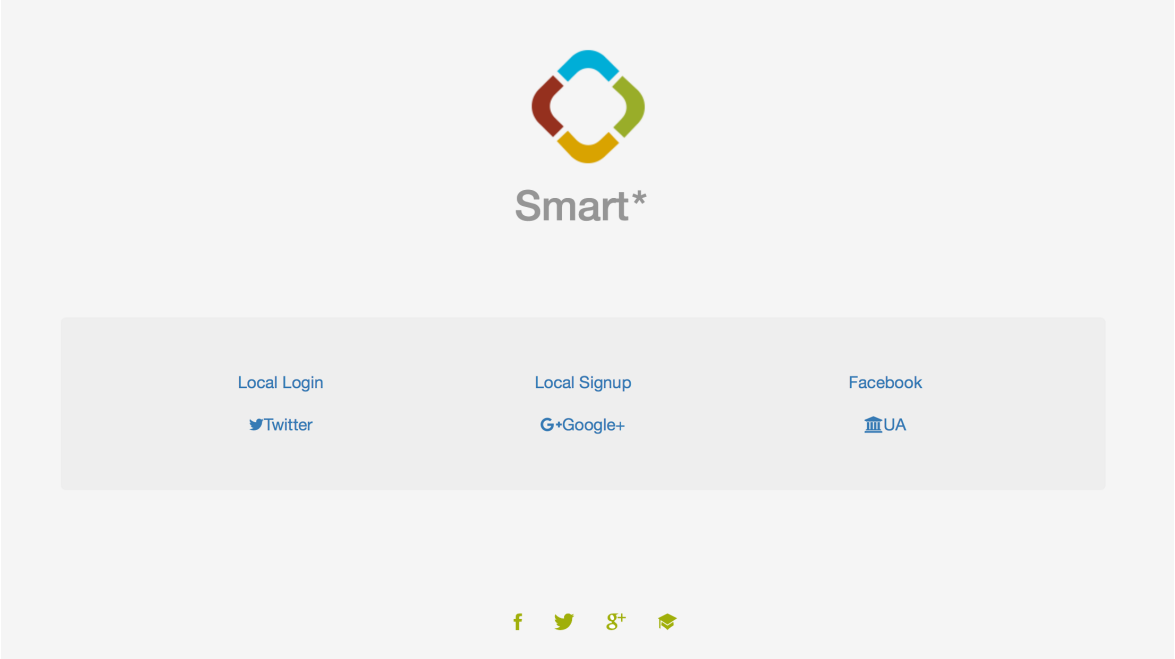

Figura 2: Ecrã inicial

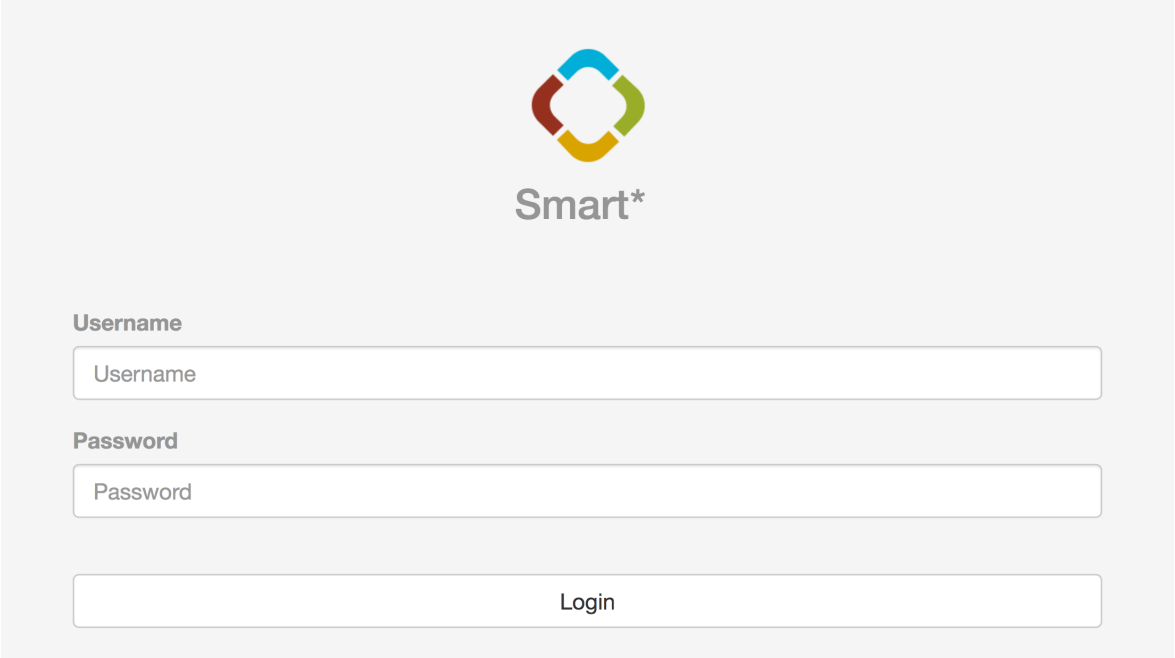

Figura 3: Ecrã de login

A partir do ecrã de boas-vindas, é possível aceder ao menu situado à direita (Figura 5), permite editar os dados do utilizador com sessão iniciada, gerir o módulo de *Flow Manager* assim como efetuar o *logout*.

A partir do ecrã de boas-vindas é possível aceder ao módulo de *Data Visualization* e ao módulo de gestão do protótipo através da *System, Device, Sensor Registration*.

Relativamente aos módulos inicialmente apresentados é possível através da aplicação criar

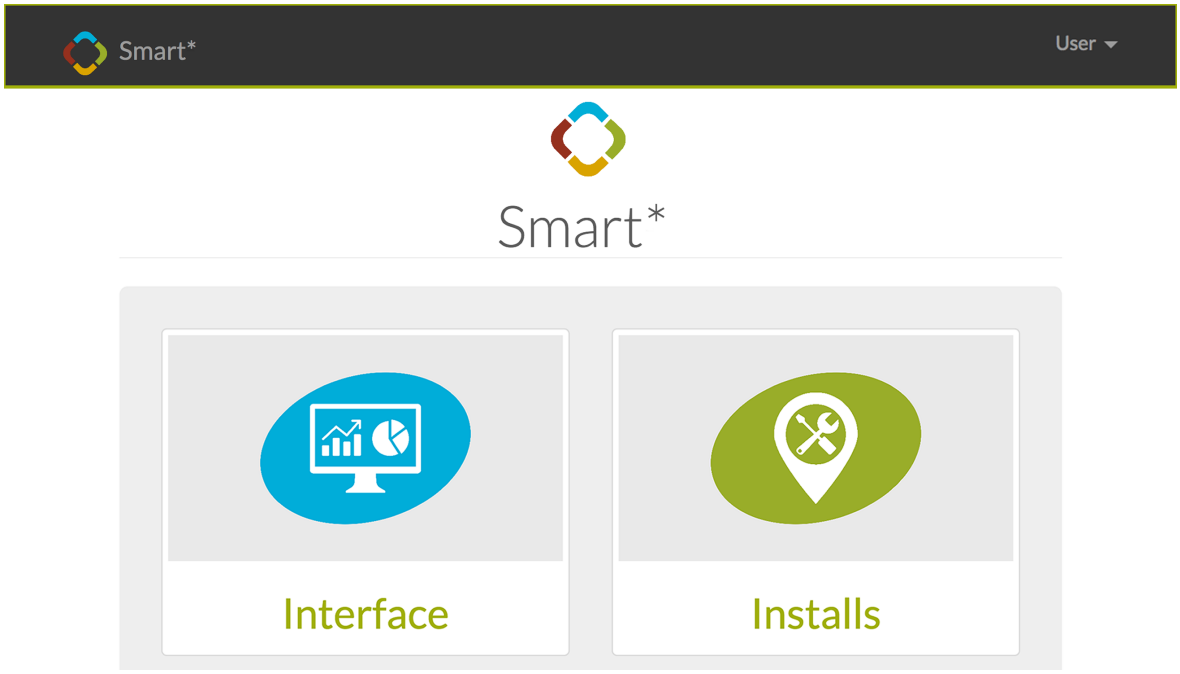

Figura 4: Ecrã de boas-vindas

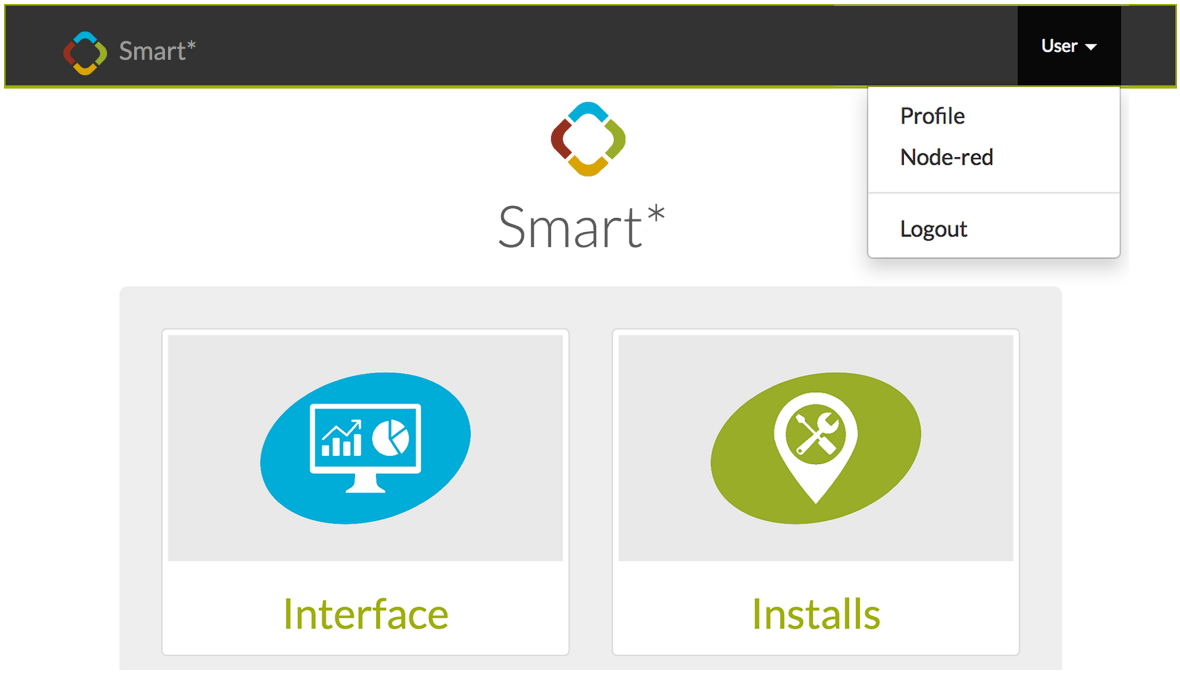

Figura 5: Menu lateral

novos cenários e instalações e fazer a gestão dos mesmos. (Figura 6) A gestão de instalações, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 6).

Para criar uma nova instalação, basta escolher um nome, seguidamente dar-lhe um tipo de instalação, indicar uma descrição e a sua localização bem como a sua distribuição geográfica. (Figura 7)

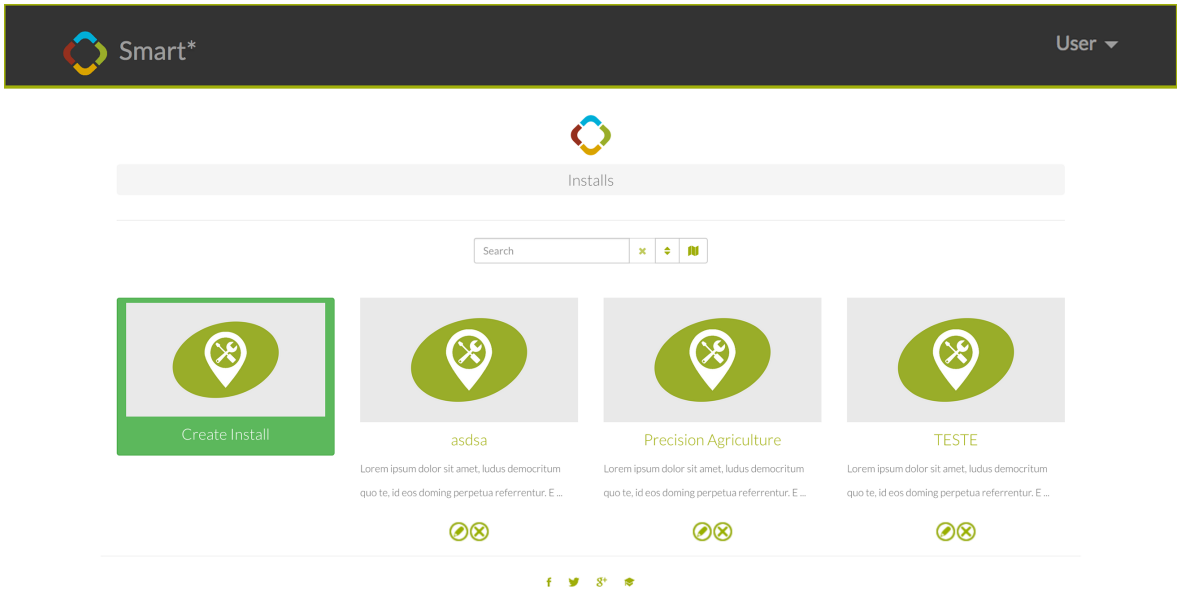

Figura 6: Lista de instalações

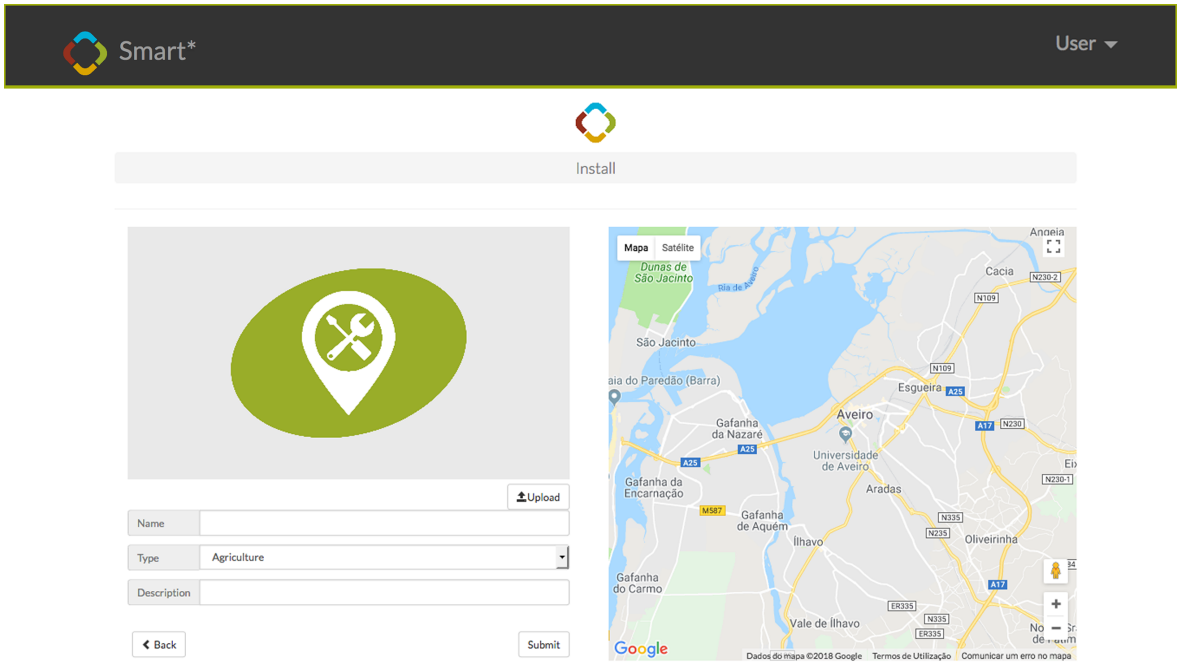

Figura 7: Adicionar nova instalação

Já na instalação anteriormente criada pelo utilizador, existe a possibilidade de visualizar as suas características bem como inserir novos equipamentos. (Figura 8).

É possível visualizar os vários equipamentos presentes numa determinada instalação. A gestão de equipamentos, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 9).

É também possível criar uma nova instalação, bastando escolher um nome, indicar uma

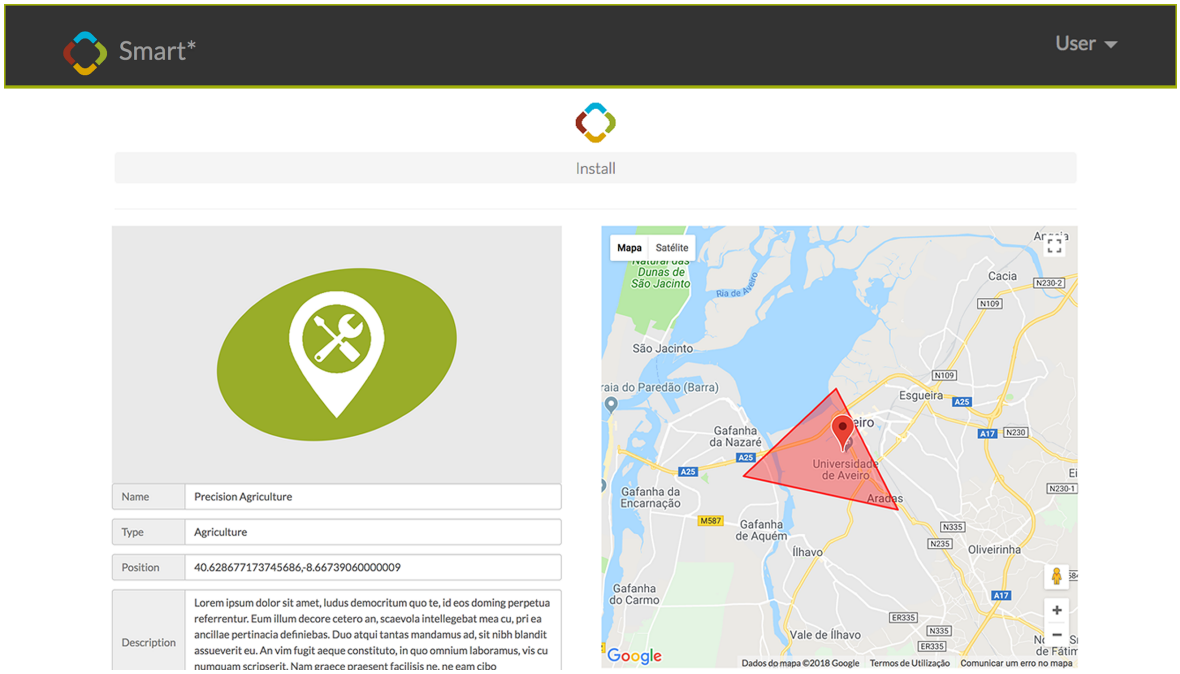

Figura 8: Visualização de instalação

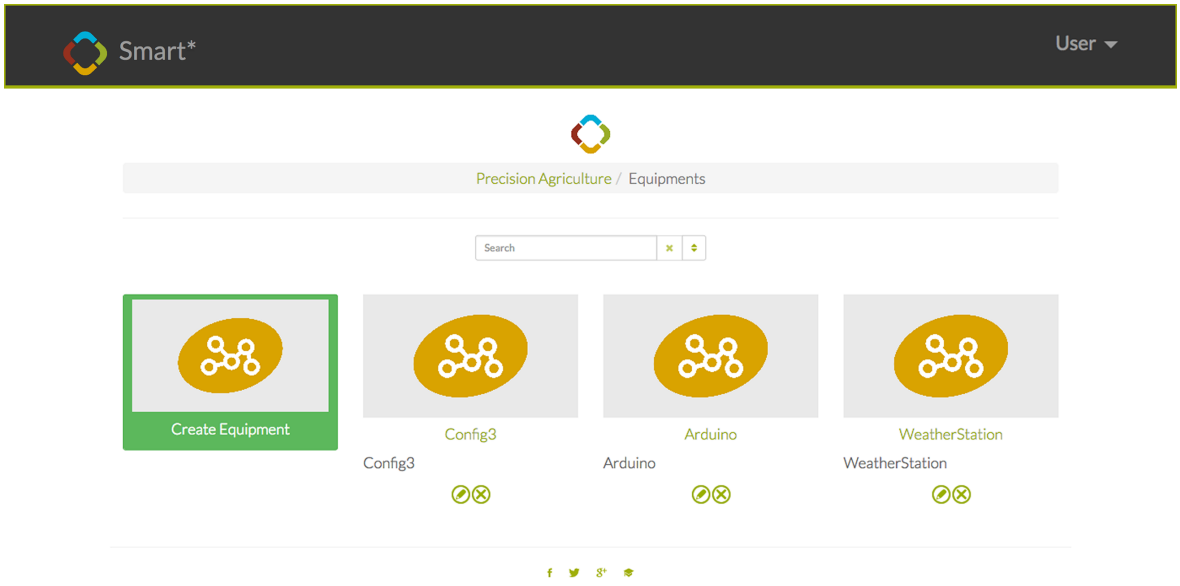

Figura 9: Lista de equipamentos presentes numa instalação

descrição e a sua localização. (Figura 10)

Ao visualizar um determinado equipamento, existe a possibilidade de visualizar as suas características bem como associar sensores ao equipamento em questão. (Figura 11).

Visualizando os sensores presentes num determinado equipamento é possível a sua gestão, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 12).

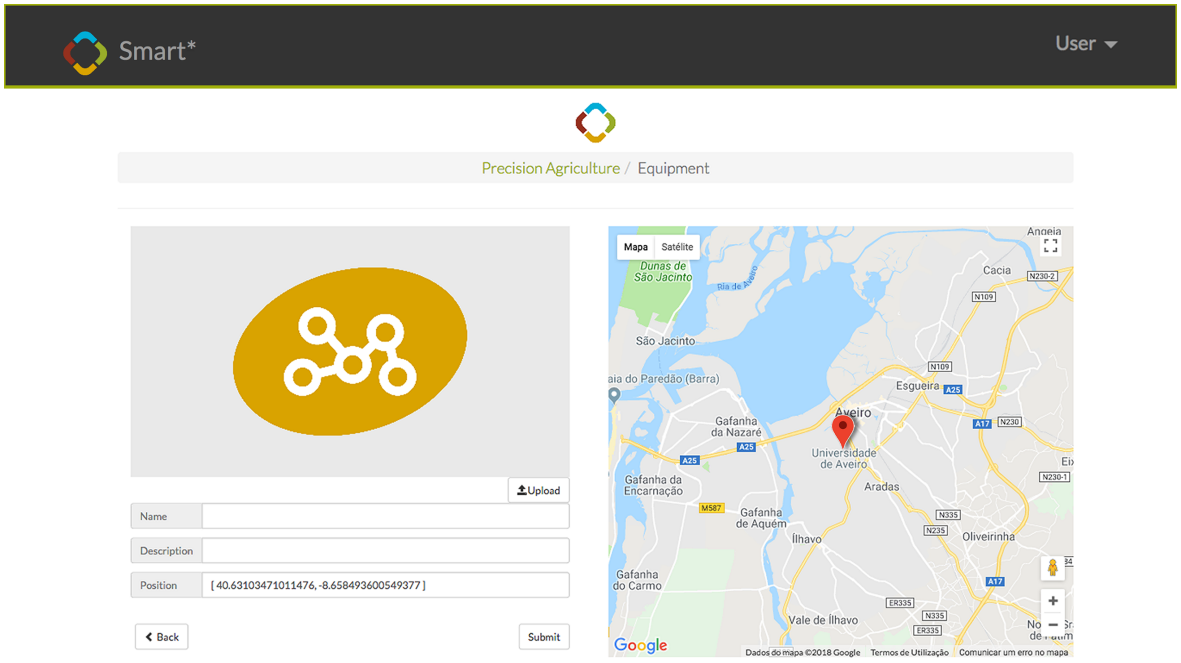

Figura 10: Adicionar novo equipmento

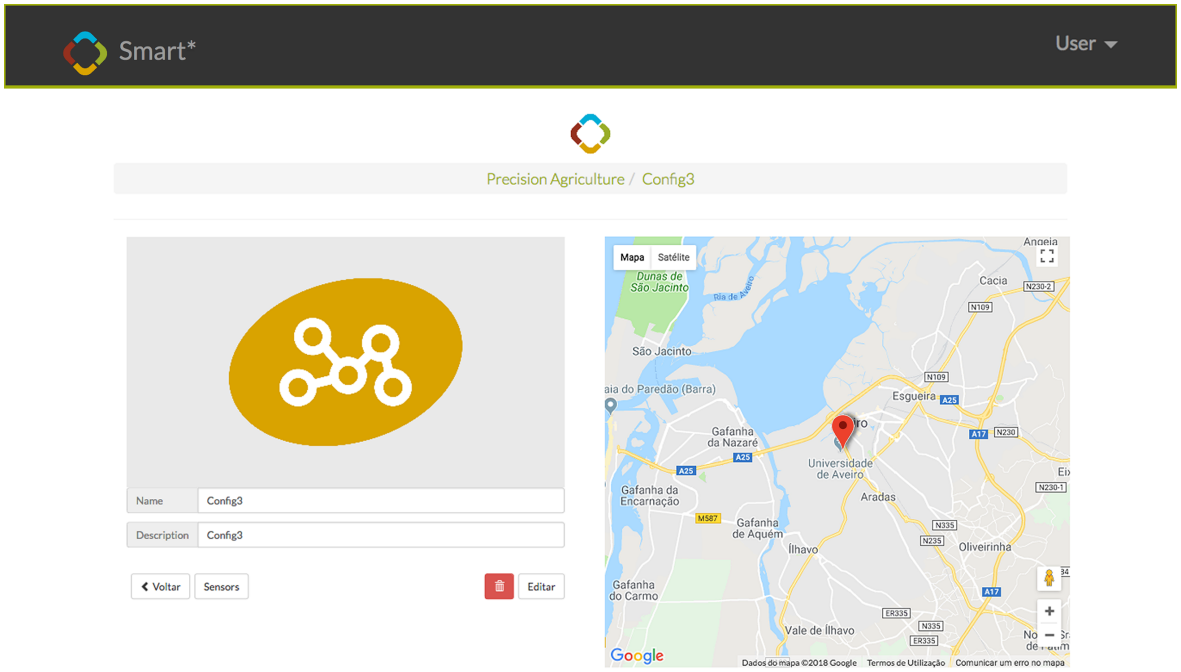

Figura 11: Visualização de equipmento

É também possível criar um novo sensor, bastando escolher um nome, indicar uma descrição, indicar o tipo de dados bem como indicar a representação pretendida no módulo de *Data Visualization*. (Figura 13)

Ao visualizar um determinado sensor, existe a possibilidade de visualizar as suas características. (Figura 14).

Tal como foi indicado anteriormente, a partir do ecrã de boas-vindas é possível aceder ao

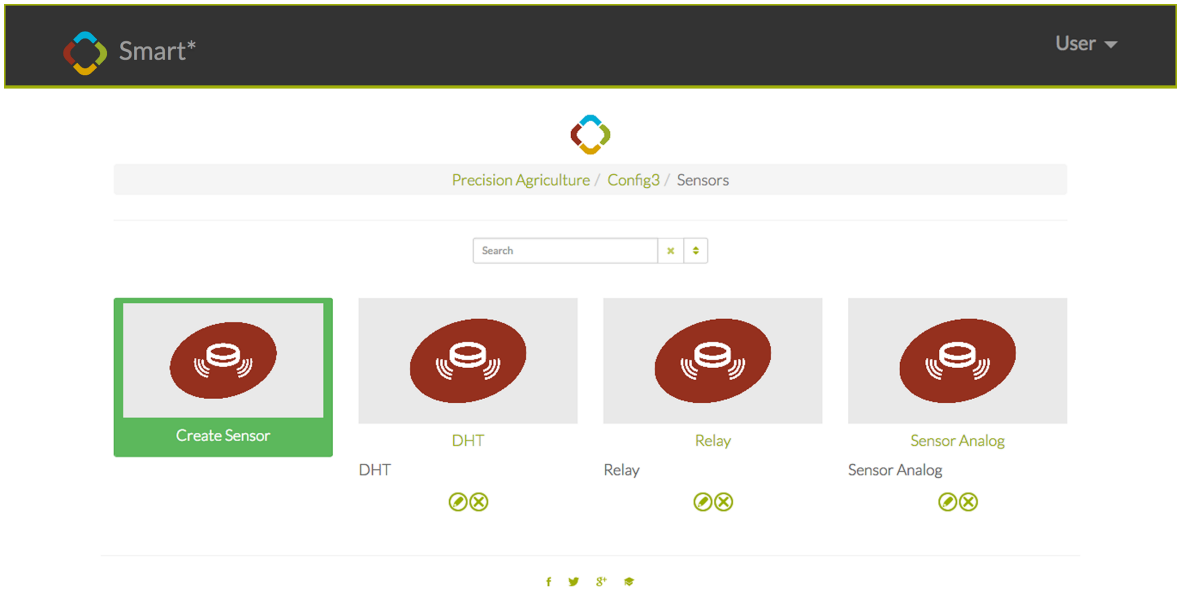

Figura 12: Lista de sensores presentes num equipamento

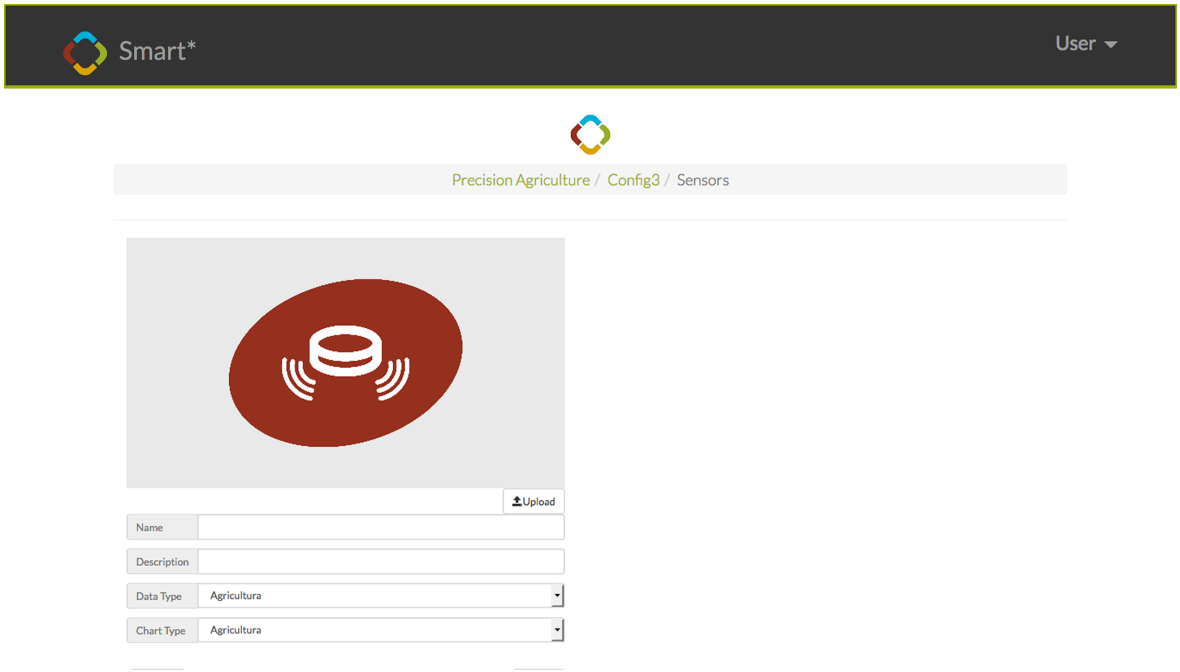

Figura 13: Adicionar novo sensor

módulo de *Data Visualization* onde é possível ao utilizador visualizar os dados recolhidos a partir dos sensores presentes na *Sensor Network*. (Figura 15)

Esta é uma visualização criada dinamicamente a partir dos dados recolhidos. Através do *Flow Manager* é possível ao utilizador personalizar a sua visualização. (Figura 16)

De seguida serão apresentado os vários ecrãs do protótipo através de um *Tablet*.

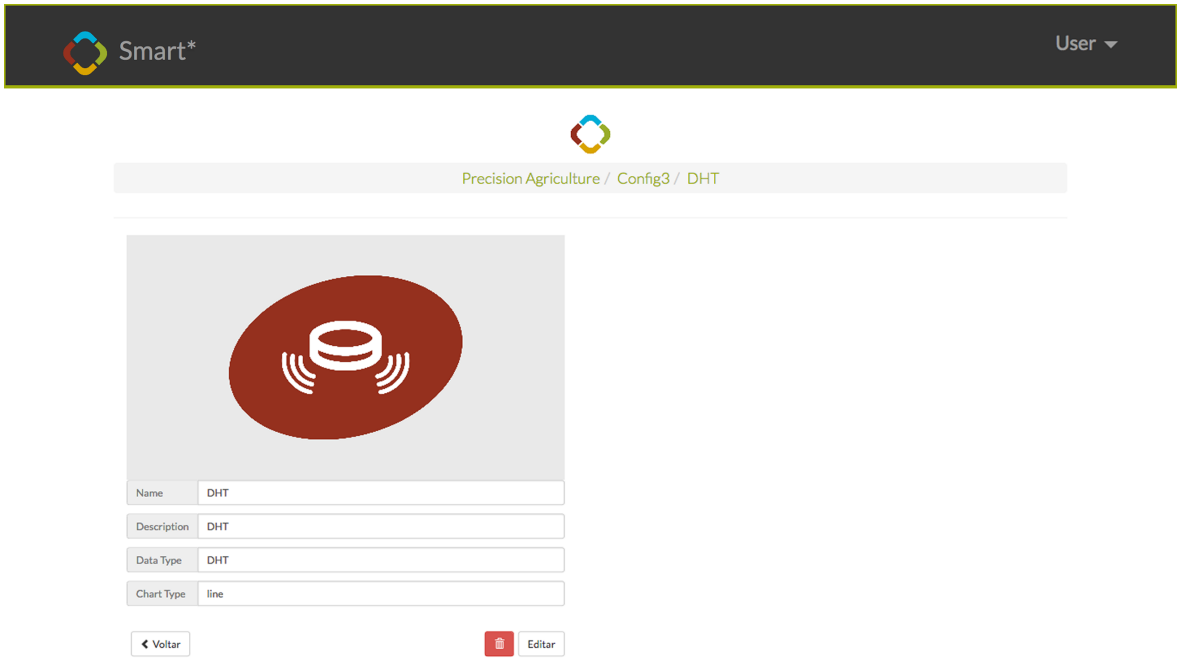

Figura 14: Visualização de sensor

Figura 15: Interface

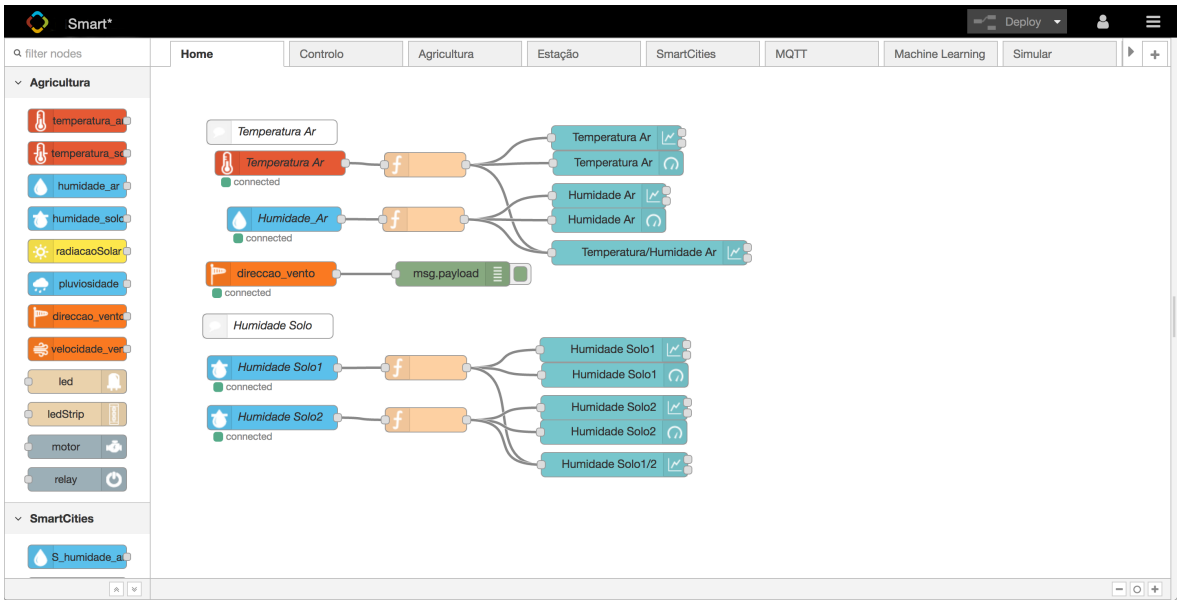

Figura 16: Orquestrador

Protótipo Tablet

As imagens seguintes têm por objetivo demonstrar as funcionalidades do sistema tendo em conta a proposta de arquitetura apresentada.

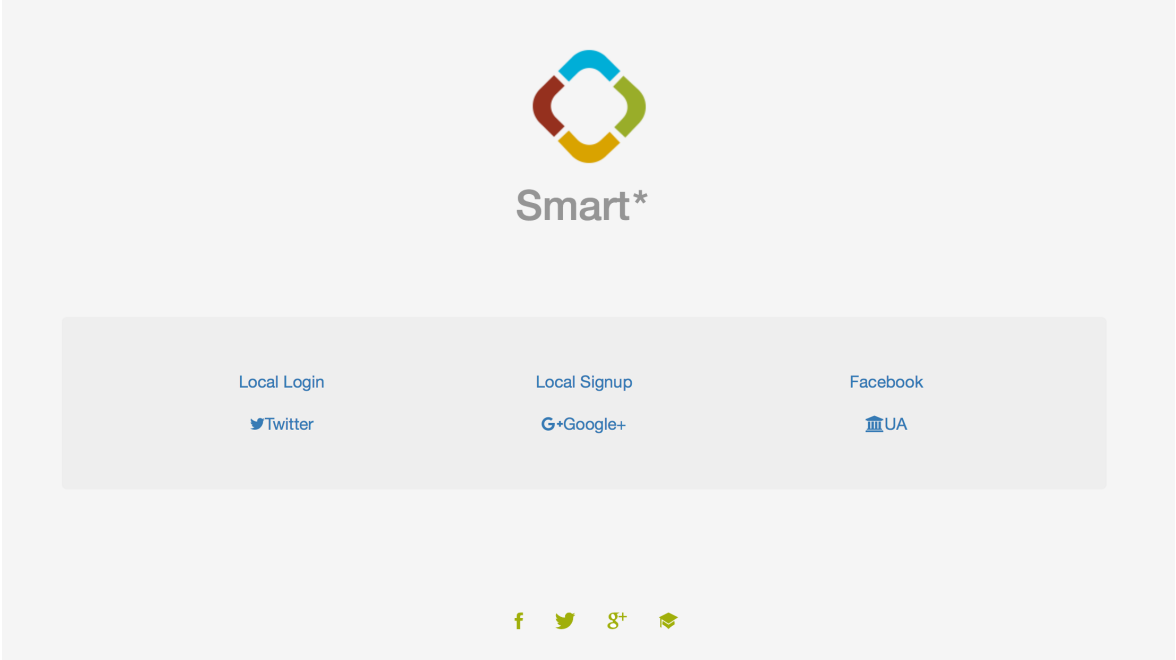

Figura 17: Ecrã inicial - Tablet

Quando a aplicação é iniciada, é mostrado ao utilizador um ecrã inicial (Figura 17) onde é possível iniciar a aplicação realizando a sua autenticação através das redes sociais para além da autenticação utilizando o par utilizador/*password* (Figura 18).

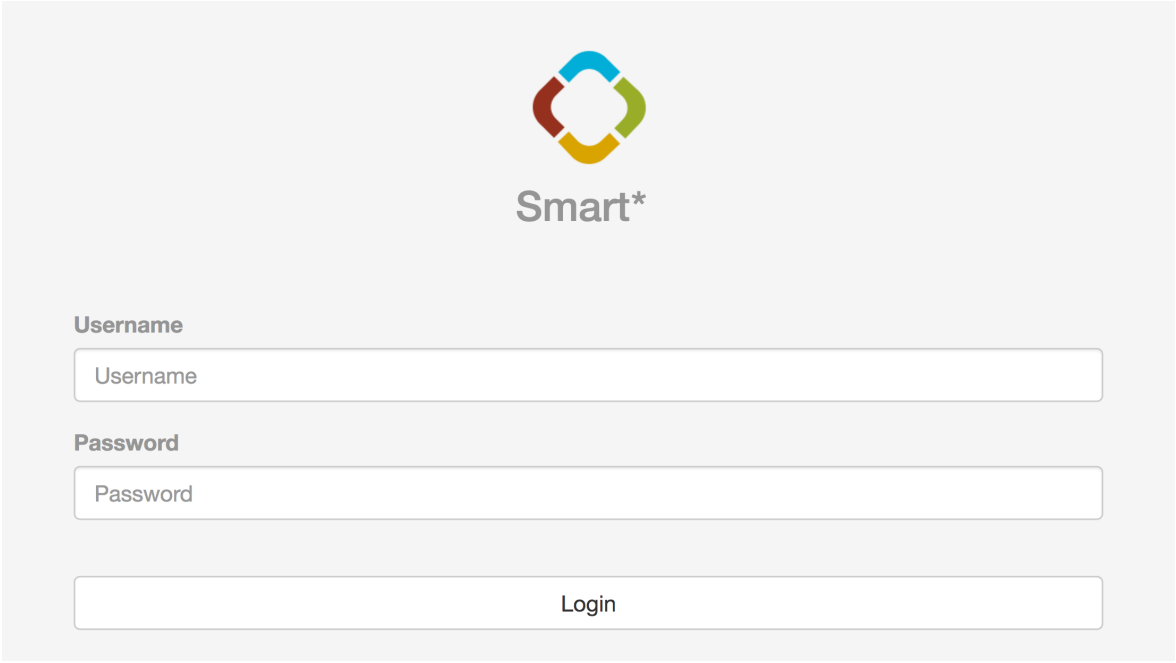

Figura 18: Ecrã de login - Tablet

Aqui, o utilizador insere as credenciais de acesso e após verificação das mesmas, o utilizador terá acesso então às funcionalidades da aplicação (Figura 19).

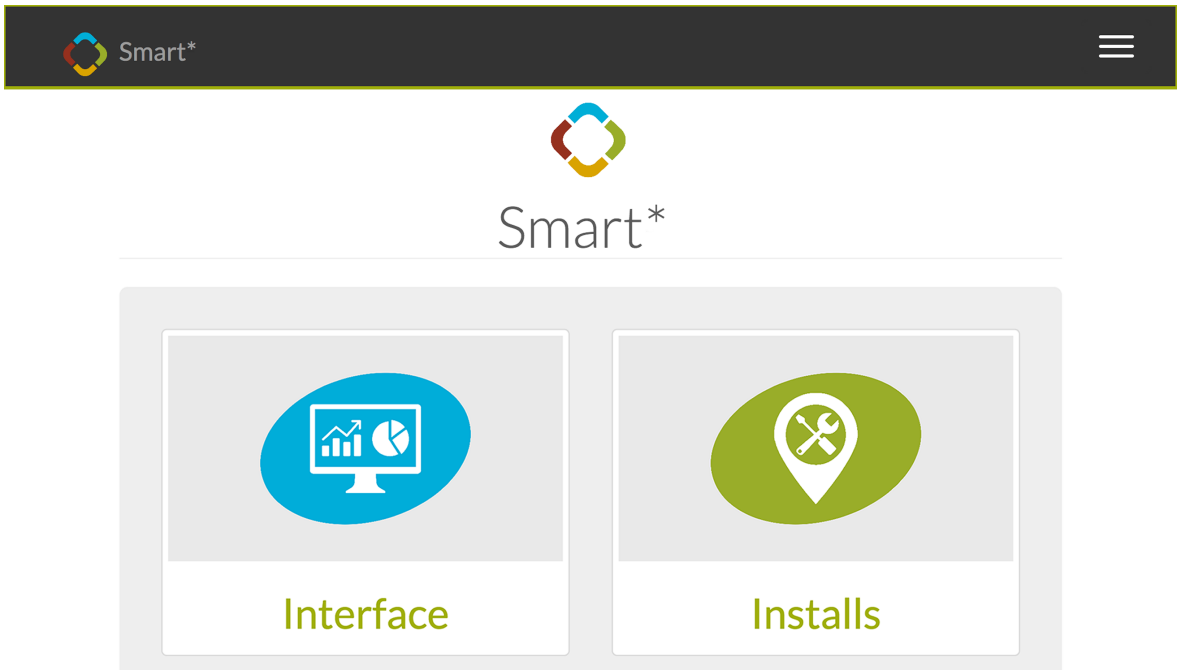

Figura 19: Ecrã de boas-vindas - Tablet

A partir do ecrã de boas-vindas, é possível aceder ao menu situado à direita (Figura 20), permite editar os dados do utilizador com sessão iniciada, gerir o módulo de *Flow Manager* assim como efetuar o *logout*.

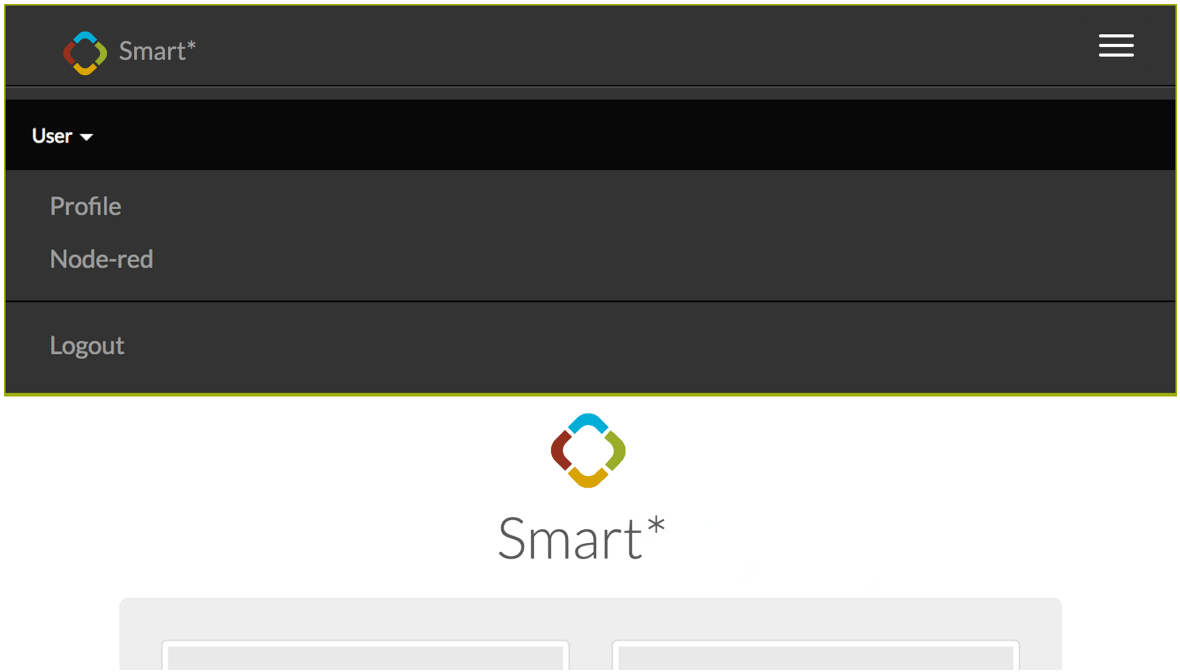

Figura 20: Menu lateral - Tablet

A partir do ecrã de boas-vindas é possível aceder ao módulo de *Data Visualization* e ao módulo de gestão do protótipo através da *System, Device, Sensor Registration*.

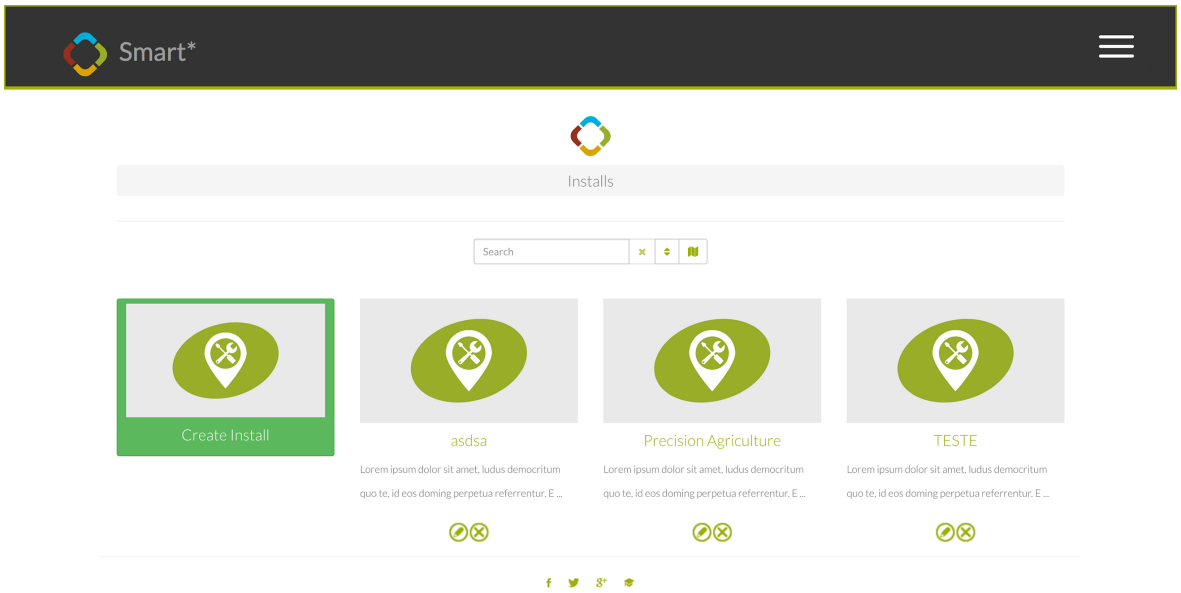

Figura 21: Lista de instalações - Tablet

Relativamente aos módulos inicialmente apresentados é possível através da aplicação criar novos cenários e instalações e fazer a gestão dos mesmos. (Figura 21) A gestão de instalações, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 21).

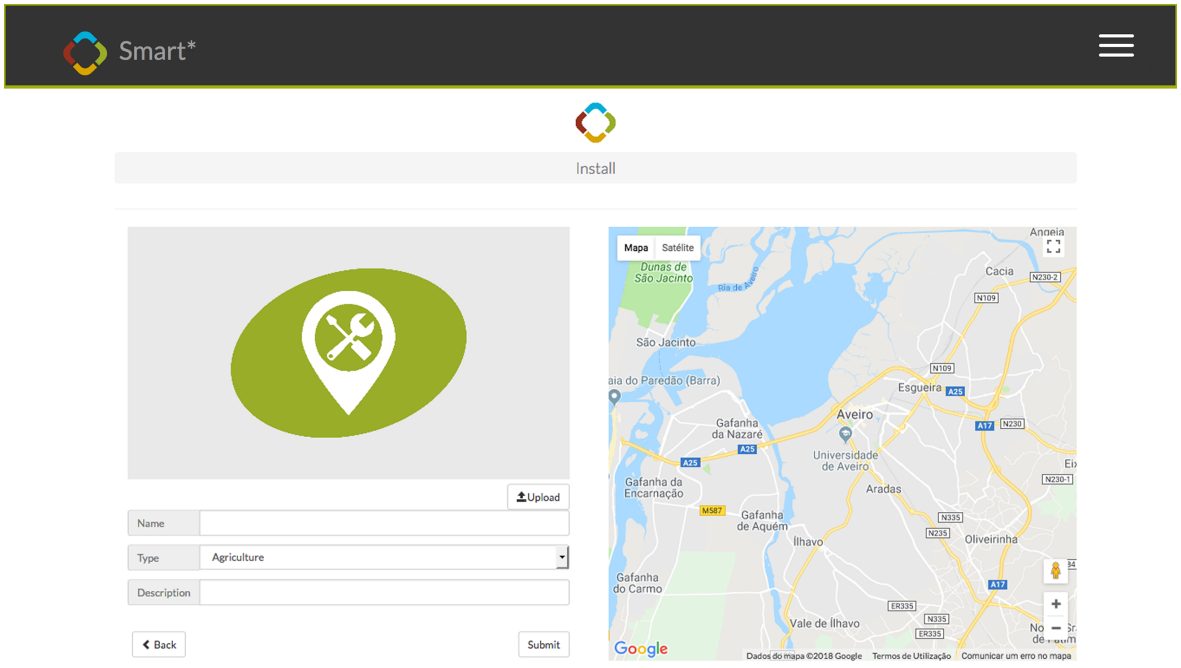

Figura 22: Adicionar nova instalação - Tablet

Para criar uma nova instalação, basta escolher um nome, seguidamente dar-lhe um tipo de instalação, indicar uma descrição e a sua localização bem como a sua distribuição geográfica. (Figura 22)

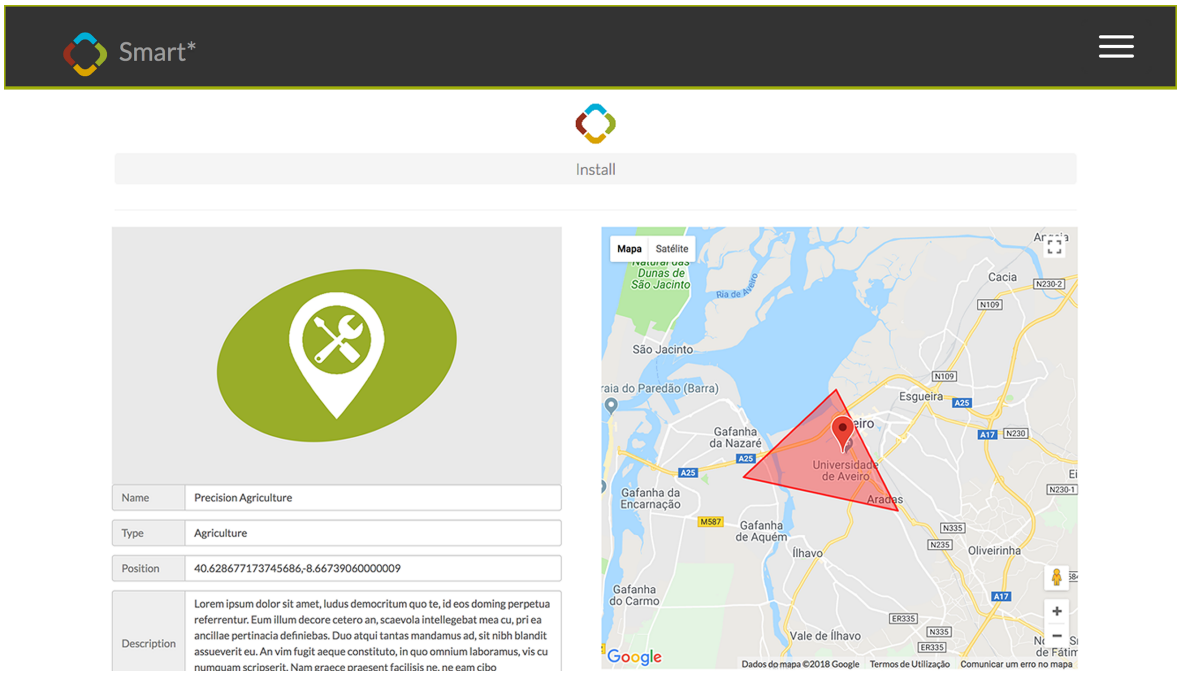

Figura 23: Visualização de instalação - Tablet

Já na instalação anteriormente criada pelo utilizador, existe a possibilidade de visualizar as suas características bem como inserir novos equipamentos. (Figura 23).

É possível visualizar os vários equipamentos presentes numa determinada instalação. A gestão de equipamentos, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 24).

É também possível criar uma nova instalação, bastando escolher um nome, indicar uma descrição e a sua localização. (Figura 25)

Ao visualizar um determinado equipamento, existe a possibilidade de visualizar as suas características bem como associar sensores ao equipamento em questão. (Figura 26).

Visualizando os sensores presentes num determinado equipamento é possível a sua gestão, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 27).

É também possível criar um novo sensor, bastando escolher um nome, indicar uma descrição, indicar o tipo de dados bem como indicar a representação pretendida no módulo de *Data Visualization*. (Figura 28)

Ao visualizar um determinado sensor, existe a possibilidade de visualizar as suas características. (Figura 29).

Tal como foi indicado anteriormente, a partir do ecrã de boas-vindas é possível aceder ao módulo de *Data Visualization* onde é possível ao utilizador visualizar os dados recolhidos a partir dos sensores presentes na *Sensor Network*. (Figura 30)

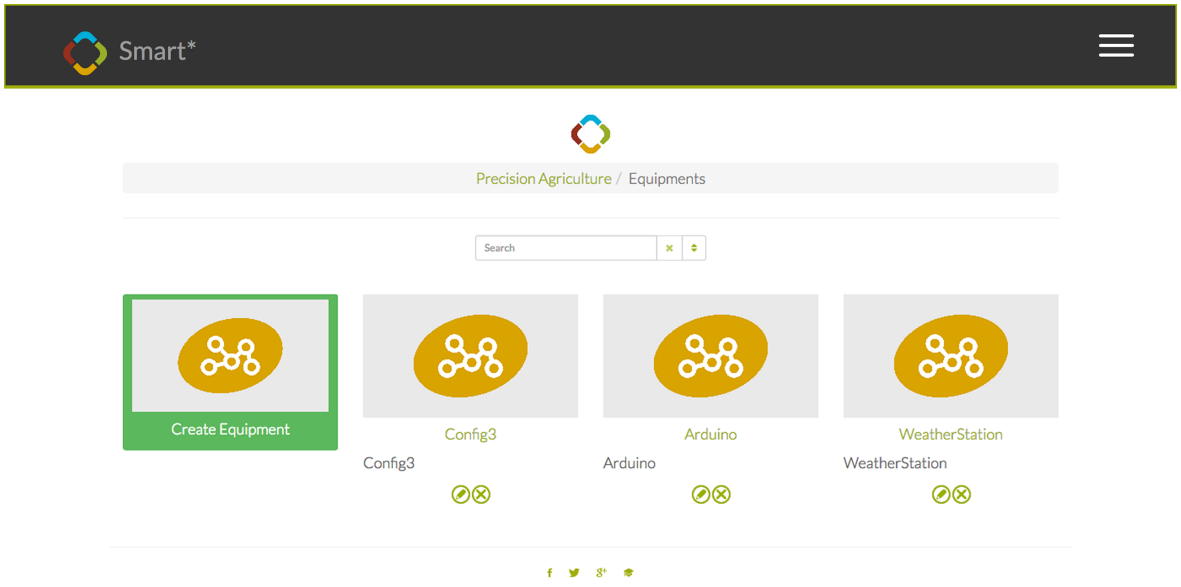

Figura 24: Lista de equipamentos presentes numa instalação - Tablet

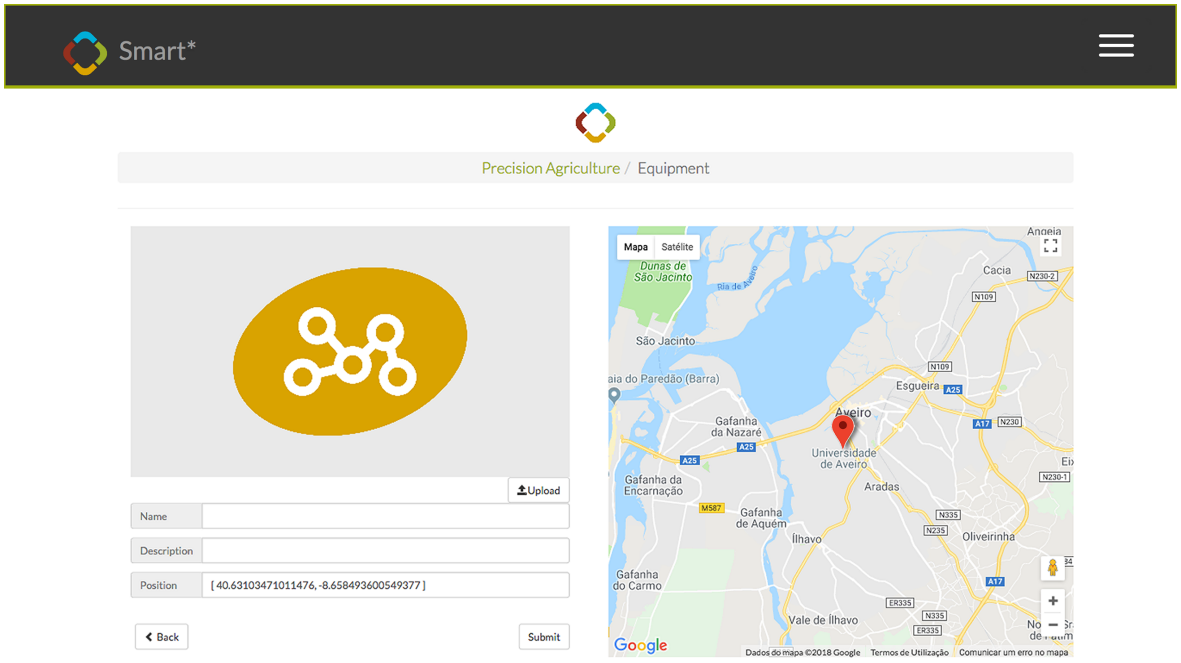

Figura 25: Adicionar novo equipamento - Tablet

Esta é uma visualização criada dinamicamente a partir dos dados recolhidos. Através do *Flow Manager* é possível ao utilizador personalizar a sua visualização. (Figura 31)

De seguida serão apresentado os vários ecrãs do protótipo através de um *Smartphone*.

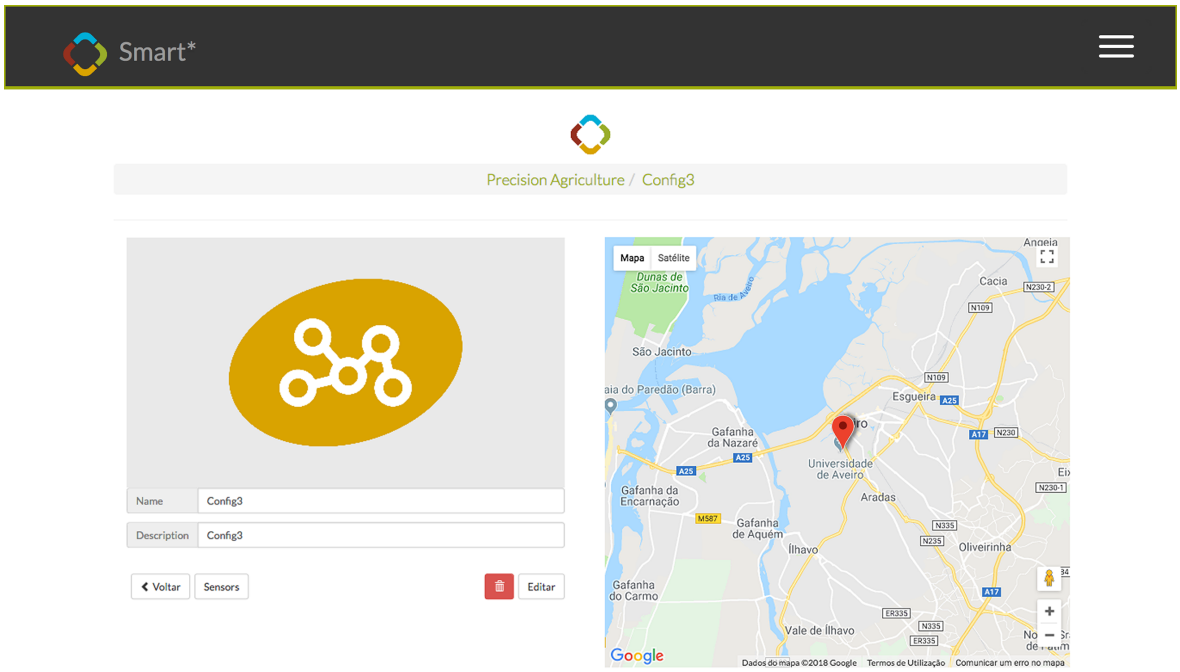

Figura 26: Visualização de equipamento - Tablet

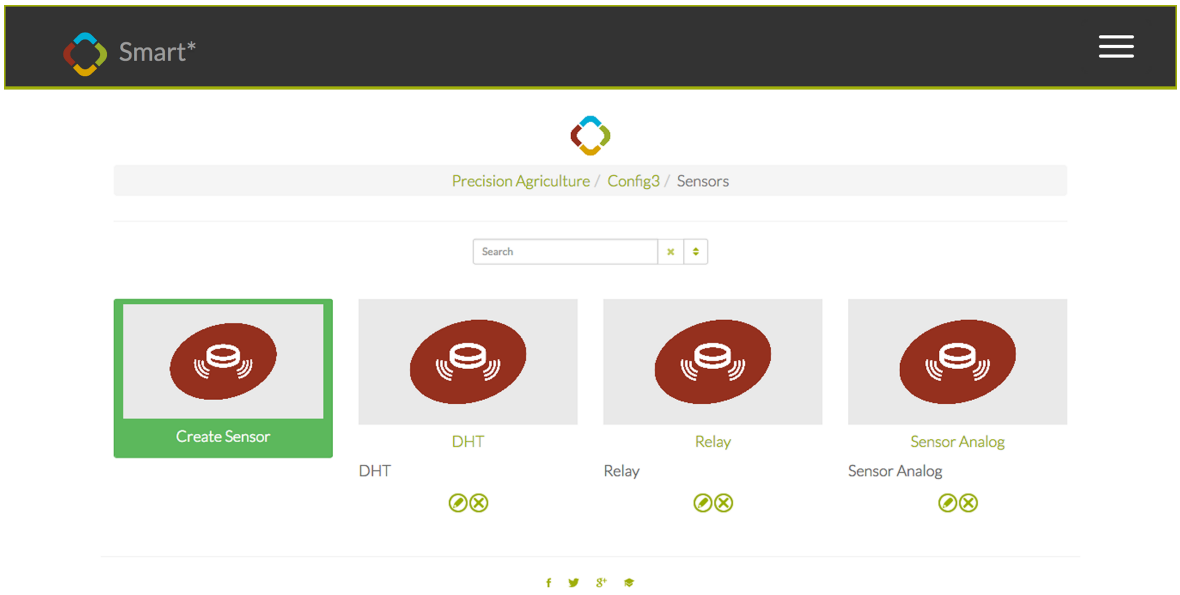

Figura 27: Lista de sensores presentes num equipamento - Tablet

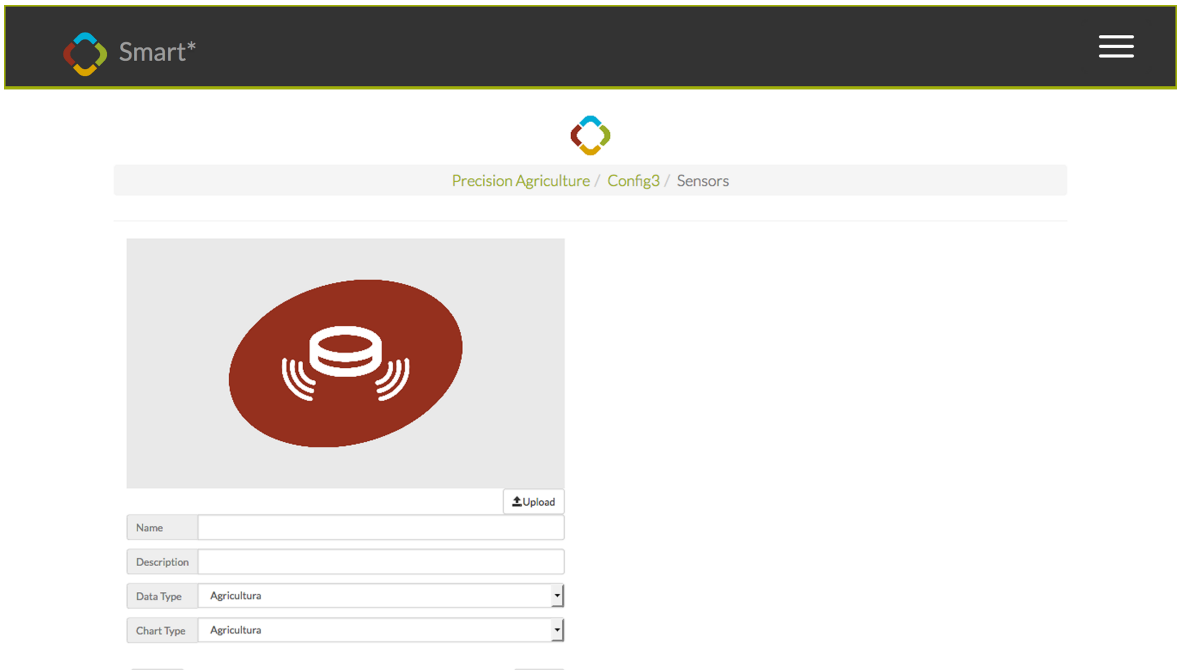

Figura 28: Adicionar novo sensor - Tablet

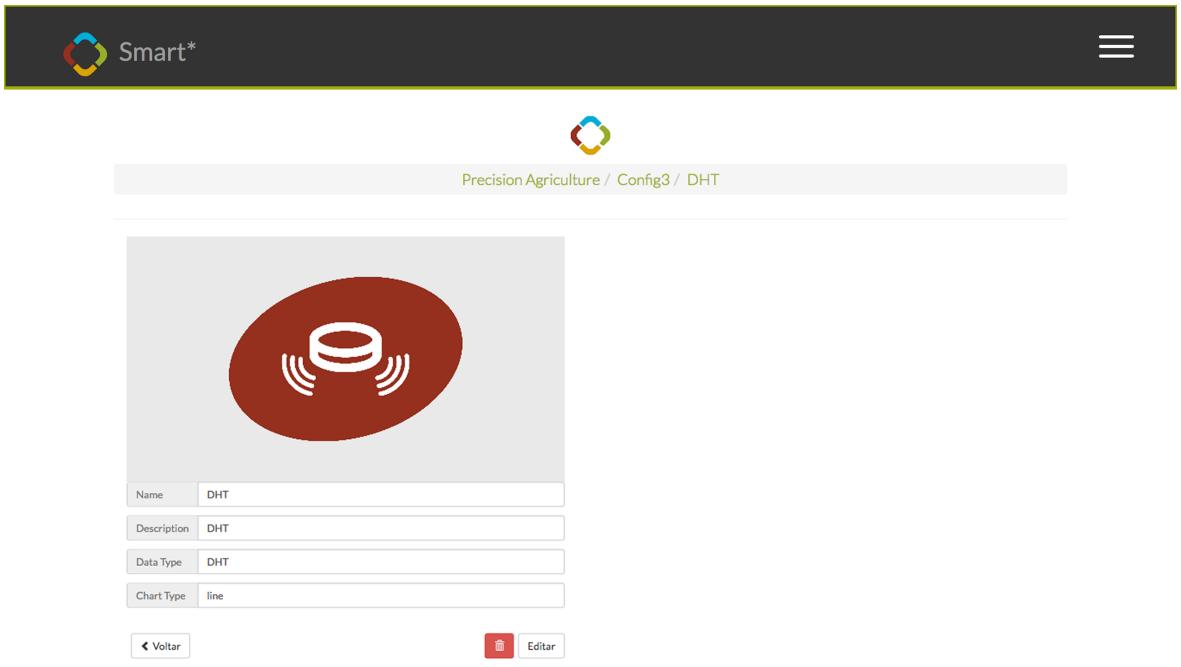

Figura 29: Visualizar sensor - Tablet

Figura 30: Interface - Tablet

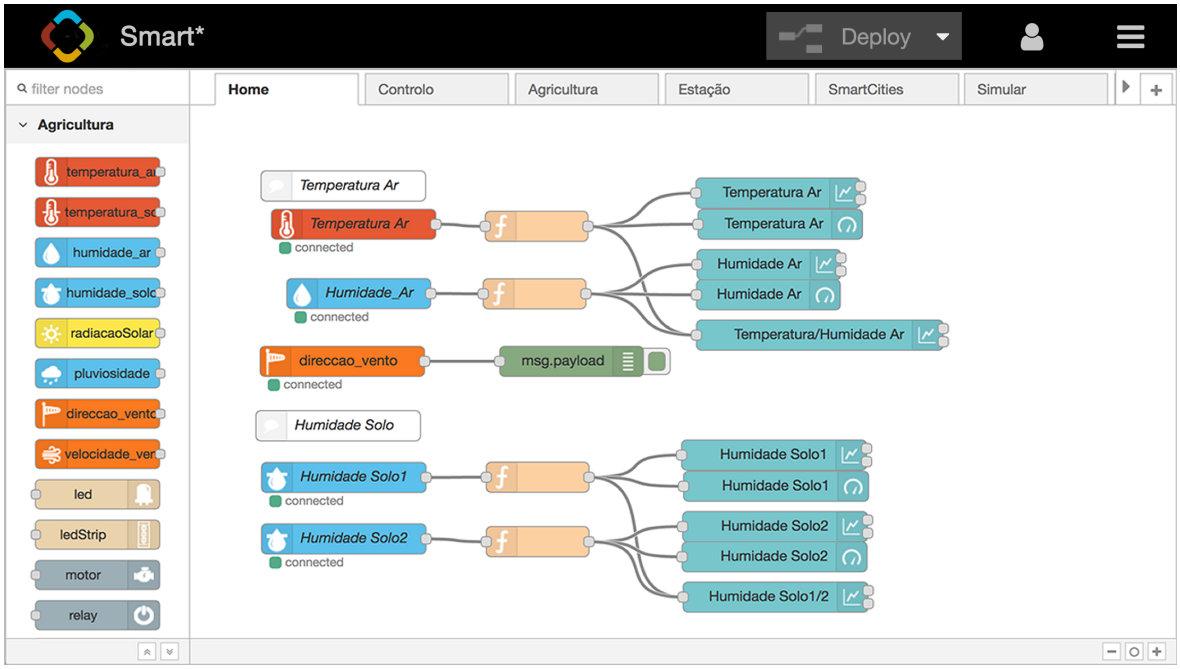

Figura 31: Orquestrador - Tablet

Protótipo Mobile

As imagens seguintes têm por objetivo demonstrar as funcionalidades do sistema tendo em conta a proposta de arquitetura apresentada.

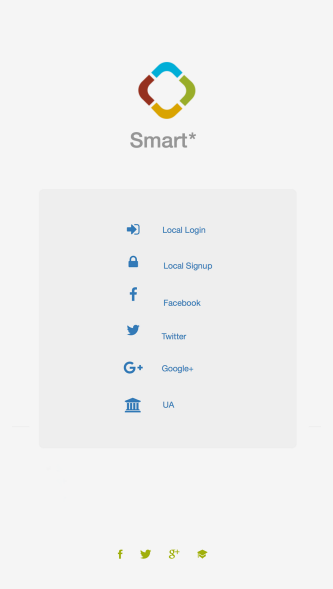

Figura 32: Ecrã inicial - Mobile

Quando a aplicação é iniciada, é mostrado ao utilizador um ecrã inicial (Figura 32) onde é possível iniciar a aplicação realizando a sua autenticação através das redes sociais para além da autenticação utilizando o par utilizador/*password* (Figura 33).

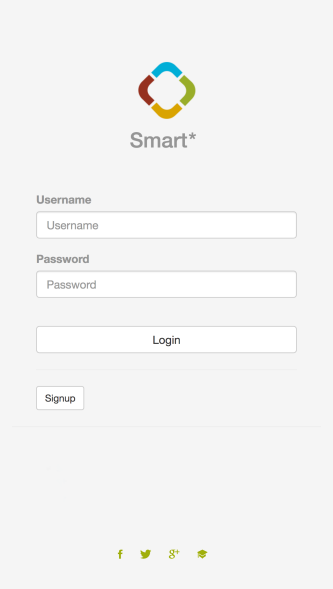

Figura 33: Ecrã de login - Mobile

Aqui, o utilizador insere as credenciais de acesso e após verificação das mesmas, o utilizador terá acesso então às funcionalidades da aplicação (Figura 34).

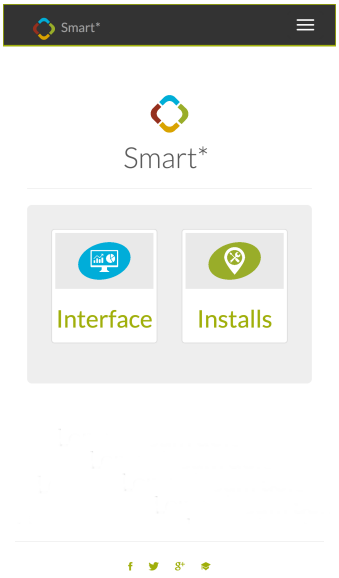

Figura 34: Ecrã de boas-vindas - Mobile

A partir do ecrã de boas-vindas, é possível aceder ao menu situado à direita (Figura 35), permite editar os dados do utilizador com sessão iniciada, gerir o módulo de *Flow Manager* assim como efetuar o *logout*.

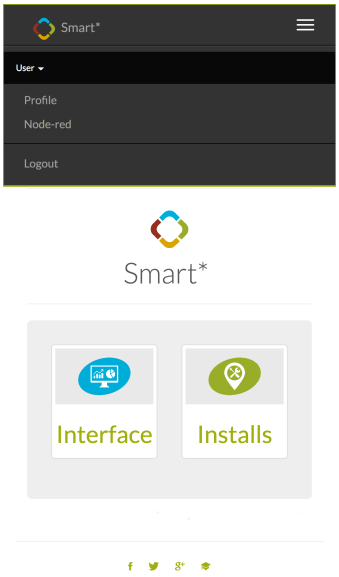

Figura 35: Menu lateral - Mobile

A partir do ecrã de boas-vindas é possível aceder ao módulo de *Data Visualization* e ao módulo de gestão do protótipo através da *System, Device, Sensor Registration.*

Relativamente aos módulos inicialmente apresentados é possível através da aplicação criar novos cenários e instalações e fazer a gestão dos mesmos. (Figura 36) A gestão de instalações, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 36).

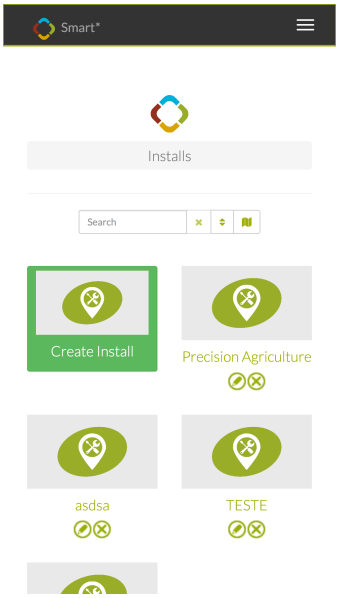

Figura 36: Lista de instalações - Mobile

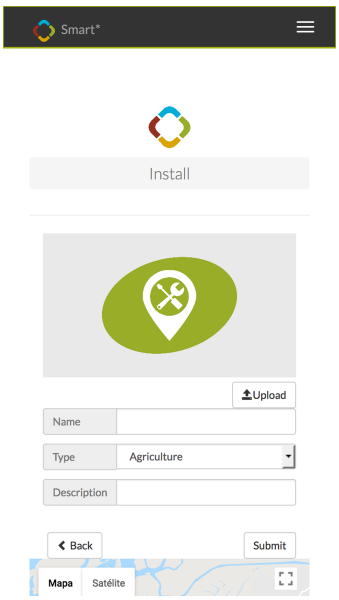

Figura 37: Adicionar nova instalação - Mobile

Para criar uma nova instalação, basta escolher um nome, seguidamente dar-lhe um tipo de instalação, indicar uma descrição e a sua localização bem como a sua distribuição geográfica. (Figura 37)

Já na instalação anteriormente criada pelo utilizador, existe a possibilidade de visualizar as suas características bem como inserir novos equipamentos. (Figura 38).

É possível visualizar os vários equipamentos presentes numa determinada instalação. A gestão de equipamentos, para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 39).

É também possível criar uma nova instalação, bastando escolher um nome, indicar uma

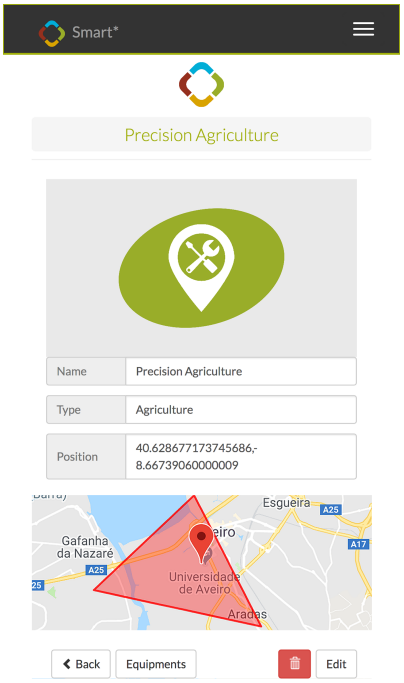

Figura 38: Visualização de instalação - Mobile

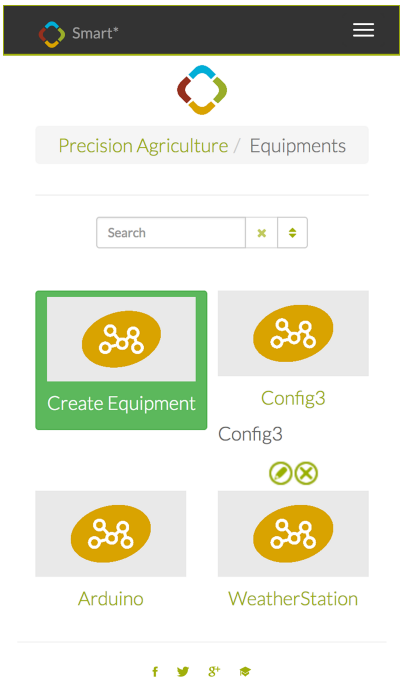

Figura 39: Lista de equipamentos presentes numa instalação - Mobile

descrição e a sua localização. (Figura 40)

Ao visualizar um determinado equipamento, existe a possibilidade de visualizar as suas características bem como associar sensores ao equipamento em questão. (Figura 41).

Visualizando os sensores presentes num determinado equipamento é possível a sua gestão,

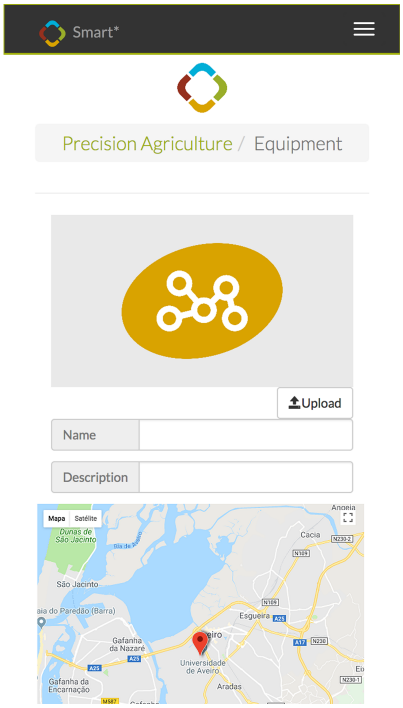

Figura 40: Adicionar novo equipamento - Mobile

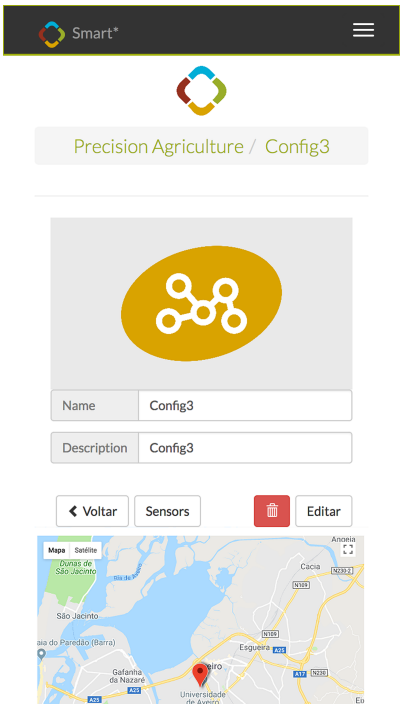

Figura 41: Visualização de equipamento - Mobile

para além de permitir obter uma listagem dos mesmos, permite ainda editar os dados ou eliminá-los caso seja necessário (Figura 42).

É também possível criar um novo sensor, bastando escolher um nome, indicar uma descrição, indicar o tipo de dados bem como indicar a representação pretendida no módulo de

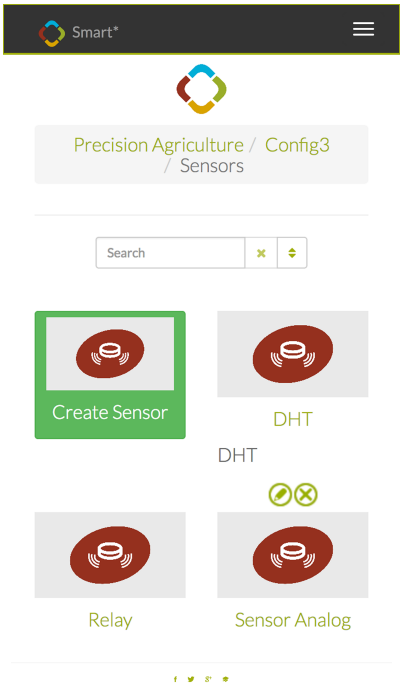

Figura 42: Lista de sensores presentes num equipamento - Mobile

Data Visualization. (Figura 43)

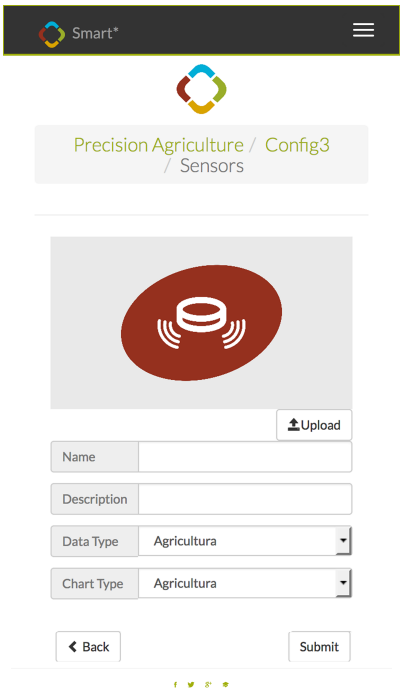

Figura 43: Adicionar novo sensor - Mobile

Ao visualizar um determinado sensor, existe a possibilidade de visualizar as suas características. (Figura 44).

Tal como foi indicado anteriormente, a partir do ecrã de boas-vindas é possível aceder ao

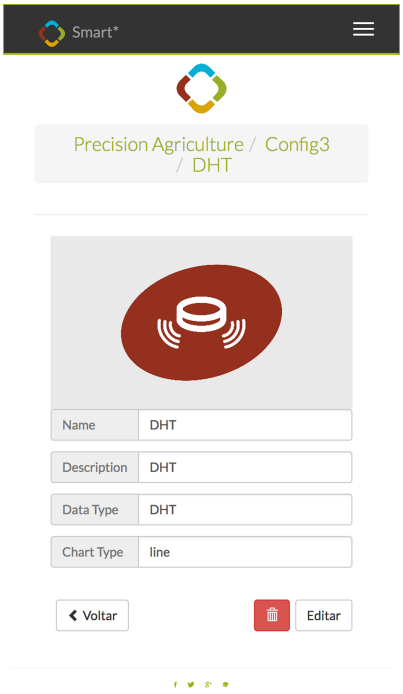

Figura 44: Visualização de sensor - Mobile

módulo de *Data Visualization* onde é possível ao utilizador visualizar os dados recolhidos a partir dos sensores presentes na *Sensor Network*. (Figura 45)

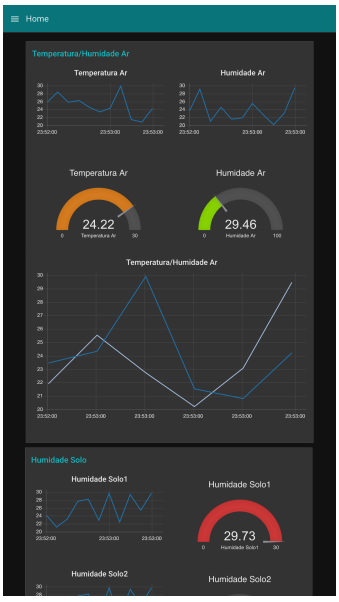

Figura 45: Interface - Mobile

Esta é uma visualização criada dinamicamente a partir dos dados recolhidos. Através do *Flow Manager* é possível ao utilizador personalizar a sua visualização. (Figura 46)

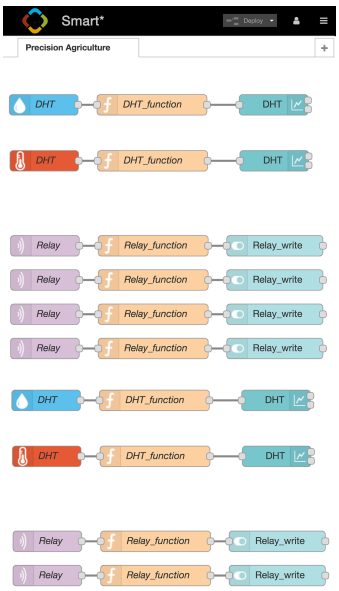

Figura 46: Orquestrador - Mobile